ICT-based innovation and its competitive outcome: the role of information intensity

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electronics Engineering (29^{th}cycle)

Design and Optimization of Graph Transform for Image and Video Compression

By

Giulia Fracastoro

******

Supervisor:
Prof. Enrico Magli

Doctoral Examination Committee:
Prof. Riccardo Bernardini
Prof. Gene Cheung
Prof. Lorenzo Galleani
Prof. Maurizio Martina
Dr. Chiara Ravazzi

Politecnico di Torino
2017

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Giulia Fracastoro
2017

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To my mother

Acknowledgements

Firstly, I would like to thank my thesis advisor, Prof. Enrico Magli, for his support and guidance during all these years. I would like to thank him for always encouraging my research and for allowing me to grow in this field. Beside my advisor, I also would like to express my deep gratitude to Prof. Pascal Frossard, who gave me the opportunity to join his lab for a visiting period, during which I have learned so many things.

During my PhD I had the oppurtunity to work closely with a number of people who contributed significantly to this thesis. I am grateful to Dr. Dorina Thanou for her help during my visiting period at EPFL. Her support was really valuable and she always encouraged me to be more optimistic about my work. I would like also to thank Prof. Marco Grangetto and Francesco Verdoja for giving me the chance of collaborating with them.

My sincere thanks also goes to all the people of Sisvel Technology for giving me this opportunity and for being helpful and supportive during my PhD.

I also would like to thank all the people from LTS4 that I have met during my visiting period in Lausanne.

A huge thanks goes to all the former and current members of IPL. They have all helped me during these years. In particular, I am grateful to my officemate Matteo for his endless patience, always listening to all my concerns. A special thanks also goes to Sophie and Chiara for always encouraging me in my research work.

Finally, I am also deeply grateful to my family and to Gianluca for their unconditional support during all these years.

Abstract

The main contribution of this thesis is the introduction of new methods for designing adaptive transforms for image and video compression. Exploiting graph signal processing techniques, we develop new graph construction methods targeted for image and video compression applications. In this way, we obtain a graph that is, at the same time, a good representation of the image and easy to transmit to the decoder. To do so, we investigate different research directions. First, we propose a new method for graph construction that employs innovative edge metrics, quantization and edge prediction techniques. Then, we propose to use a graph learning approach and we introduce a new graph learning algorithm targeted for image compression that defines the connectivities between pixels by taking into consideration the coding of the image signal and the graph topology in rate-distortion term. Moreover, we also present a new superpixel-driven graph transform that uses clusters of superpixel as coding blocks and then computes the graph transform inside each region.

In the second part of this work, we exploit graphs to design directional transforms. In fact, an efficient representation of the image directional information is extremely important in order to obtain high performance image and video coding. In this thesis, we present a new directional transform, called Steerable Discrete Cosine Transform (SDCT). This new transform can be obtained by steering the 2D-DCT basis in any chosen direction. Moreover, we can also use more complex steering patterns than a single pure rotation. In order to show the advantages of the SDCT, we present a few image and video compression methods based on this new directional transform. The obtained results show that the SDCT can be efficiently applied to image and video compression and it outperforms the classical DCT and other directional transforms. Along the same lines, we present also a new generalization of the DFT, called Steerable DFT (SDFT). Differently from the SDCT, the SDFT can be defined in one

or two dimensions. The 1D-SDFT represents a rotation in the complex plane, instead the 2D-SDFT performs a rotation in the 2D Euclidean space.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In image and video compression, the two dimensional discrete cosine transform (DCT) is by far the most widespread transform [1]. The DCT is used in the majority of image and video compression standards, such as JPEG, H264 and HEVC. For example, in the latest video coding standard HEVC, the core transform is an integer approximation of the DCT [2]. Also other common video codecs, such as Thor [3] and VP9 [4], use an integer approximation of the DCT for residual coding.

The DCT can efficiently represent smooth signals, but it becomes inefficient when the signal contains large discontinuities. In this case, the DCT will generate a non-sparse signal representation and the high-frequency coefficients can have large magnitude. This will result in poor coding performance. In such cases, an adaptive transform, which adapts its basis functions to the signal structure, may provide better performance.

In this thesis, we investigate two different types of adaptive transforms. In the first case, we use a graph-based approach. Any image can be viewed as a graph, where each pixel is a node of the graph and the edges describe the connectivity relations among pixels, e.g. in terms of similarity [5]. It is possible to define a transform on this domain, called graph Fourier transform (GFT) [5]. Thanks to the graph representation, the corresponding transform is "aware" of image discontinuities, which are downplayed so as to minimize generation of high-frequency coefficients and maximize energy compaction.

In the second case, we exploit graph signal processing techniques in order to propose a new directional transform. This new directional transform, called steerable discrete cosine transform (SDCT), can be obtained by steering the 2D-DCT basis functions in any chosen direction. In this way, we incorporate directional information into the 2D-DCT obtaining a transform that follows the dominant direction of the image and better preserves the directional information of the image.

In the next two sections we describe more in detail these two approaches.

1.1 Graph-based image and video compression

In the last few years, researchers have made some attempts to perform image and video compression using graph signal processing techniques. Block-based methods using graph Fourier transform have been proposed in [6–8], but they reported unsatisfactory results on natural images that are not piecewise smooth. For the specific case of residual coding, a few methods based on GFT have been recently proposed. A novel graph-based method for intra-frame video coding has been presented in [9], which introduces a new generalized graph Fourier transform optimized for intra-prediction residues. Instead, in [10] the authors propose a block-based lifting transform on graphs for intra-predicted video coding. Moreover, a graph-based method for inter-predicted video coding has been introduced in [11], where the authors design a set of simplified graph templates capturing basic statistical characteristics of inter-predicted residual blocks. Another graph-based method for inter-predicted video coding has been recently proposed in [12], where the authors propose a new class of graph-based transforms, called symmetric line graph transforms.

Another approach to graph-based image and video compression is presented in [13], where the authors introduce a new lifting-based wavelet transform on graphs. In [14, 15], a novel approach for image compression using graph-based wavelet filterbanks is presented. Instead, in [16–18] the authors propose a complete video encoder based on lifting-based wavelet transforms on graphs. They construct a graph in which any pixel could be linked to a number of spatial and temporal neighbors, in this way they exploit jointly spatial and temporal correlation.

One of the main drawbacks of graph-based compression techniques lies in the cost required to represent and encode the graph, which may outweigh the coding gain provided by the edge adaptive transform. In order to solve this problem, in [8] the authors propose to use a lookup table where they store the most commonly used graph transforms, and then during encoding they just transmit the index of the corresponding chosen transform. However, this method present several issues when applied to larger blocks (such as 16×16 or 32×32 blocks) due to the increasing number of possible transforms. This will lead to a very large lookup table, that is difficult to handle both from the memory and computational side. In our work we focus on developing new graph construction techniques for graph-based compression, obtaining a graph that at the same time gives an efficient image representation and is not too complex. In this thesis we introduce the following techniques:

- a new method of graph construction for image compression applications that employs innovative edge metrics, quantization and edge prediction techniques.
- a novel graph learning algorithm targeted for image compression that uncovers the connectivities between pixels by taking into consideration the coding of the image signal and the graph topology in rate-distortion term, obtaining in this way a graph that optimizes the overall rate-distortion performance.
- a new superpixel-driven graph transform that uses clusters of superpixels [19], which have the ability to adhere nicely to edges in the image, as coding blocks and computes inside these homogeneously colored regions a graph transform which is shape adaptive.

1.2 Directional transform

Efficient representation of directional information is extremely important for high performance image and video coding [20]. For this reason, recently several directional transforms have been proposed and applied in image and video coding. Most of these transforms consist in modification of the 2D-DCT in order to incorporate directional information [21–25]. The directional DCT (DDCT)

presented in [21] is the first attempt in this sense. It consists in a separable transform in which the first 1D-DCT may follow a direction other than the vertical or the horizontal one; then the coefficients produced by all directional transforms in the first step are rearranged so that the second transform can be applied to those coefficients that are best aligned with each other. Later, other works have followed this approach. In [22], the authors have introduced new directions for the first transform and have proposed a new zigzag scanning method. In [23], it is suggested to not apply the second-stage DCT, or to apply it only on the DC coefficients generated during the first transform [24]. In [25], DDCT [21] is improved using anisotropic local basis supports, where the optimal basis is selected exploiting the bintree structure of the dictionary.

These methods, however, have several issues. In particular, they require 1D-DCT of various lengths, some of which are very short and are not always a power of 2; moreover, the second DCT may not always be applied to coefficients of similar AC frequencies [26]. In our tests, we have also noticed that the performance of the DDCT decreases when the block size increases.

Another method to introduce directionality in the DCT has been presented in [27], where directional primary operations have been introduced for the lifting-based DCT. In this way, the DCT-like lifting transform can be applied along any direction, but it extends across block boundaries in order to apply direction adaptation.

In the specific case of intra-frame video coding, another approach has been investigated: the transform is constructed by a directional prediction and a corresponding data-dependent transform. In [28], mode-dependent directional transforms have been derived from Karhunen-Loève transform, using prediction residuals from training video data. Various follow-up works have then enhanced [28] exploiting the symmetry to reduce the number of transform matrices needed [29–31]. To further improve the performance, several other mode-dependent directional transforms have been proposed, such as the mode-dependent sparse transform [32] and the rate-distortion optimized transform [33]. Another data-dependent directional transform called Sparse Orthonormal Transform has been proposed in [34] and [35]. In this case, the image blocks are classified using the image gradient. Then, the transform of each class is optimized by minimizing on a training set an approximation cost. A common problem of these methods

is that training sets must be processed to obtain transforms that are optimal for a given class, so the transform is always dependent on the training set used.

In this thesis, we present a new framework for directional transforms. Starting from the graph transform of a grid graph, we design a new transform, called steerable DCT (SDCT), which can be obtained by steering the 2D-DCT basis in a chosen direction. The proposed transform can be oriented in any possible direction and we can use more complex steering patterns than a single pure rotation. We also show that the SDCT can be applied to image and video compression problems providing a significant quality gain.

Moreover, we also introduce a novel generalization of the DFT, called steerable DFT (SDFT). Differently from the SDCT, the SDFT can be defined in one or two dimensions. The 1D-SDFT can be interpreted as a rotation of the basis vectors in the complex plane. Instead, the 2D-SDFT represents a rotation on the two-dimensional Euclidean space.

1.3 Thesis organization

The reminder of this thesis is organized as follows.

In Chapter 2 we introduce the graph signal processing framework. In particular, we focus on graph-based image processing techniques. Chapter 3 describes a novel graph costruction technique targeted for image compression. We introduce a new method for defining the edge weights of the graph and efficiently coding them. In particular, we introduce a new method of edge prediction that nearly halves the cost of graph transmission.

In Chapter 4, we propose a novel framework for designing the graph in image compression applications. Using a graph learning approach, we choose the optimal graph by solving a rate-distortion optimization problem that takes into account the coding of the image as well as the cost of transmitting the graph. Moreover, we also introduce an innovative method for coding the graph by treating its weights as a graph signal that lies on the dual graph.

Chapter 5 describes an innovative image compression method that jointly employs computer vision tools and the graph Fourier transform. We first subdi-

vide the image into uniform regions that adhere well to the image boundaries, then we apply a graph transform within each region.

We then move to directional transforms in Chapter 6. In this chapter we present a new directional transform called steerable DCT (SDCT). Then, in Chapter 7 we present a few image compression algorithms based on the SDCT.

Along the same line, in Chapter 8 we present a new directional DFT, called steerable DFT (SDFT).

1.4 Publications

In this section we provide a list of publications which are the outcome of the research carried out during the PhD program.

1. G. Fracastoro, E. Magli, Predictive Graph Construction for Image Compression, *Proc. of IEEE International Conference on Image Processing*, 2015, pp. 2204-2208. (Best student paper award (third place))

2. G. Fracastoro, F. Verdoja, M. Grangetto, E. Magli, Superpixel-driven Graph Transform for Image Compression, *Proc. of IEEE International Conference on Image Processing*, 2015, pp. 2631-2635.

3. G. Fracastoro, E. Magli, Steerable Discrete Cosine Transform, *Proc. of IEEE International Workshop on Multimedia Signal Processing*, 2015, pp. 1-6.

4. G. Fracastoro, E. Magli, Subspace-sparsifying Steerable Discrete Cosine Transform from Graph Fourier Transform, *Proc. of IEEE International Conference on Image Processing*, 2016, pp. 1534-1538.

5. G. Fracastoro, D. Thanou, P. Frossard, Graph Transform Learning for Image Compression, *Proc. of Picture Coding Symposium*, 2016. (Best paper award)

6. G. Fracastoro, S. M. Fosson, E. Magli, Steerable Discrete Cosine Transform, *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 303-314, 2017.

7. G. Fracastoro, E. Magli, Steerable Discrete Fourier Transform, *accepted for publication in* IEEE Signal Processing Letters.

Chapter 2

Graph signal processing

Graphs are generic data representation forms that are useful for describing the geometric structures of data domains in numerous applications, including social, energy, trasportation, sensor, and neuronal networks [5]. Since these types of data do not have a regular domain, it is not possible to process them using the classical techniques of signal processing. For this reason, the new field of graph signal processing addresses the problem of processing signals that live on irregular domains.

A graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges with $|\mathcal{E}| = M$ and each edge connects two nodes of the graph. A graph is denoted a weighted graph if each edge has a weight value associated to it; otherwise we denote it as an unweighted graph. If we consider a weighted graph, its weight often represents the similarity between the two nodes it connects and it is possible to represent the graph by its weighted adjacency matrix $W(\mathcal{G}) \in \mathbb{R}^{N \times N}$, where, if there is an edge connecting nodes i and j, $W(\mathcal{G})_{ij}$ is the weight associated to the edge (i, j), otherwise $W(\mathcal{G})_{ij} = 0$. Instead, if we consider an unweighted graph, we can represent it by its adjacency matrix $A(\mathcal{G}) \in \mathbb{R}^{N \times N}$, where, if there is an edge connecting nodes i and j, $A(\mathcal{G})_{ij} = 1$, otherwise $A(\mathcal{G})_{ij} = 0$. In addition, a graph is said to be an undirected graph if all its edges are bidirectional, i.e. the edge (i, j) is equal to the edge (j, i). Moreover, we can say that a graph is connected if there is a sequence of edges, called path, which connects every

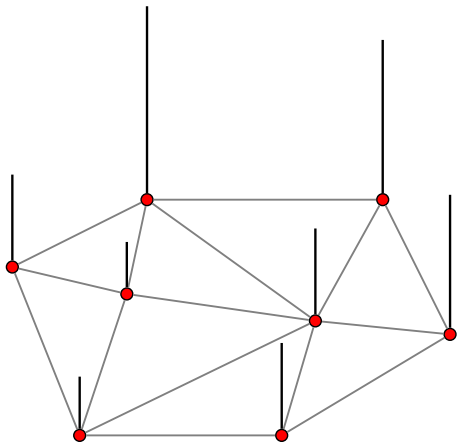


Fig. 2.1 An example of graph signal: the height of the black bar represents the value of the signal in each node.

pair of nodes of the graph. If the graph is not connected, we can identify K connected components, where $K > 1$.

In this thesis, we consider undirected graphs with no self loops (i.e. there are no edges (i,i) $\forall i \in \mathcal{V}$), thus the adjacency matrix is symmetric and has null diagonal.

We can define a graph signal $x : \mathcal{V} \to \mathbb{R}$ in the vertex domain as a real-valued function defined on the nodes of the graph $\mathcal{G}$. The graph signal may be represented as a vector $\mathbf{x} \in \mathbb{R}^N$, where the i-th component is the value of the signal at node $i \in \mathcal{V}$ [5]. An example of graph signal is shown in Fig. 2.1.

The (unnormalized) graph Laplacian, also called the combinatorial graph Laplacian, is defined as $L(\mathcal{G}) = D(\mathcal{G}) - W(\mathcal{G})$ (or $L(\mathcal{G}) = D(\mathcal{G}) - A(\mathcal{G})$ if the graph is unweighted[1]), where $D(\mathcal{G})$ is a diagonal matrix whose i-th diagonal element $D(\mathcal{G})_{ii}$ is the sum of the weights of all edges incident to node i. The graph Laplacian can be interpreted as a difference operator, because for any graph signal $\mathbf{x} \in \mathbb{R}^N$ it satisfies the following relation

$$[L(\mathcal{G})\mathbf{x}]_i = \sum_{j \in \mathcal{N}_i} W_{ij}(\mathcal{G})(x_i - x_j),$$

where $\mathcal{N}_i$ represents the neighborhood of the node i and contains all the nodes connected to i by an edge. The graph Laplacian $L(\mathcal{G})$ is also used to measure

[1]From now on, for the sake of simplicity we will consider only a weighted graph, but analogous results can be obtained for unweighted graphs.

the smoothness of a graph signal $\mathbf{f}$ on $\mathcal{G}$ [36]

$$\mathbf{x}^T L(\mathcal{G})\mathbf{x} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(\mathcal{G})(x_i - x_j)^2.$$

Since the adjacency matrix is real symmetric and $D(\mathcal{G})$ is a diagonal matrix, $L(\mathcal{G})$ is a real symmetric matrix, and therefore it is diagonalizable by an orthogonal matrix

$$L(\mathcal{G}) = \Psi\Lambda\Psi^T,$$

where $\Psi \in \mathbb{R}^{N\times N}$ is the eigenvector matrix of $L(\mathcal{G})$ that contains the eigenvectors as columns and $\Lambda \in \mathbb{R}^{N\times N}$ is the diagonal eigenvalue matrix where the eigenvalues are sorted in increasing order. The eigenvalues $\{\lambda_l\}_{l=0,\dots,N-1}$ of $L(\mathcal{G})$ are real and nonnegative. Moreover, zero is an eigenvalue of $L(\mathcal{G})$ with multiplicity equal to the number of connected components of the graph [37].

The eigenvectors of the Laplacian are used to define the graph Fourier transform (GFT) [5] of the signal $\mathbf{x}$ as follows

$$\hat{\mathbf{x}} = \Psi^T\mathbf{x}.$$

The graph signal $\mathbf{x}$ can be easily retrieved from $\hat{\mathbf{x}}$ by inversion, namely $\mathbf{x} = \Psi\hat{\mathbf{x}}$. Analogous to the Fourier transform in the Euclidean domain, the GFT is used to describe the graph signal in the Fourier domain. As shown in [5], the graph Laplacian eigenvalues and eigenvectors provide a notion of frequency. For connected graphs, the eigenvector ψ_0 associated with the eigenvalue $\lambda_0 = 0$ is constant in each node. The eigenvectors corresponding to the lowest eigenvalues are smooth on $\mathcal{G}$, i.e. if two nodes are connected by an edge with a large weight, the values of the eigenvector in these nodes are likely to be similar. Instead, the eigenvectors corresponding to larger eigenvalues oscillate more rapidly and are more likely to have dissimilar values on nodes connected by edges with large weight.

It is possible to recast some existing transforms as graph Fourier transforms on a specific topology. An example is the equivalence between the 1D-DCT and the graph Fourier transform of a path graph. We define a path graph $\mathcal{P}_N$ as a graph with N vertices and line topology, as shown in Fig. 2.2(a). It is known that the eigenvectors of $L(\mathcal{P}_N)$ are equal to the basis vectors of the

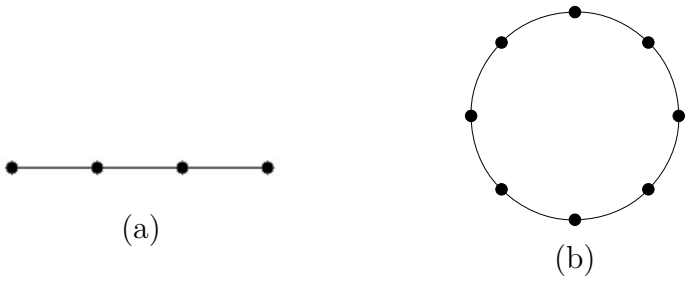


Fig. 2.2 Two graph models: (a) the path graph $\mathcal{P}_4$, (b) the cycle graph $\mathcal{C}_8$.

1D-DCT (more precisely DCT-2) [38]. Therefore, the DCT is a valid GFT of a path graph. Moreover, there is also an equivalence between the 1D-DFT and the graph Fourier transform of a cycle graph $\mathcal{C}_N$, whose structure is shown in Fig. 2.2(b). This type of graph is called a circulant graph because its adjacency matrix, and therefore its Laplacian matrix, is circulant. It is well known that a valid set of eigenvectors for any circulant matrix is the set of DFT matrix rows, then the 1D-DFT is a valid GFT for $\mathcal{C}_N$.

2.1 Graph-based image processing

Graph signal processing can be used also in classical image processing applications. An image signal can be interpreted as a graph signal where the nodes of the graph are the pixels of the image and each pixel is connected to the neighboring pixels. Usually, the edge weights between two neighboring pixels are set according to the similarity of the two pixels [39]. An example of a graph of an image block is shown in Fig. 2.3. In this way, it is possible to embed the structure of the image into the associated graph representation and one can design signal-adaptive transforms or filters. This can be useful in many image processing applications, such as compression, denoising and other image enhancement techniques.

Image compression is an important application field of graph signal processing. The graph representation of the image captures the main characteristics of the signal and allows one to design an adaptive transform in an elegant and effective way. The resulting transform is "aware" of the image discontinuities and it can lead to a very compact representation of the signal, where the signal

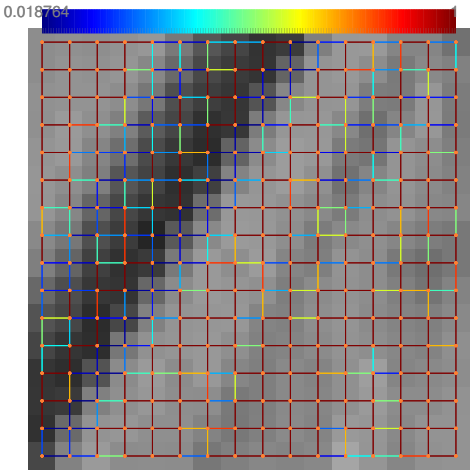


Fig. 2.3 An image block and its corresponding graph.

energy is concentrated in few low frequencies coefficients. As shown in the previous chapter, numerous graph-based image and video coding methods have been presented in the last years [6–18].

Image denoising is another important application field of graph signal processing. Since image denoising is an under-determined problem, it is important to define appropriate regularizations or priors. In [40], a GFT domain sparsity prior has been proposed. Moreover, recently some researchers propose to use the graph Laplacian regularizer as a possible prior [41–44]. The graph Laplacian regularizer states that a desirable image patch is smooth with respect to a defined graph $\mathcal{G}$. Therefore, the strength and direction of the resulting local filter depends on the edge weights that define the adjacency matrix. These methods perform competitively with other state-of-the-art methods for natural images and outperform them significantly for piecewise smooth images such as depth maps [40, 43].

Graph signal processing techniques are applied also in other inverse imaging problems. In [45], the authors use a graph-based regularization problem for upsampling low-resolution depth images. Instead, other researchers propose to use the graph Laplacian regularizer for super-resolution [46], deblurring [47] and soft decoding of JPEG images [48, 49].

In the remainder of this thesis, we will focus only on the specific problem of graph-based image compression. More precisely, we will propose new techniques

for designing the graph in order to find a good tradeoff between the cost of the graph transmission and the quality of the transform. Moreover, we also use the graph Fourier transform in order to define a new directional transform.

Chapter 3

Predictive graph construction for image compression

While graph-transforms have been shown to be more efficient than conventional transforms, the overhead of graph transmission may easily outweigh the coding efficiency benefits. Therefore, it is very important to design graph representations and corresponding graph transforms that are efficient also when graph has to be transmitted to a decoder. In this chapter, we propose a new method of graph construction for graph-based image compression, obtaining a graph that at the same time gives an efficient image representation and is not too complex. We introduce a new technique for defining the edge weights of the graph and efficiently coding them. One of the main novelty of our work is the development of a technique for edge prediction that nearly halves the cost for graph transmission. With the proposed method, we outperform existing techniques achieving an average gain of 1.6 dB in PSNR compared to the DCT transform.

Part of the work described in this chapter has been previously published in G. Fracastoro, E. Magli, Predictive Graph Construction for Image Compression, *Proc. of IEEE International Conference on Image Processing*, 2015, pp. 2204-2208.

3.1 Proposed graph construction technique

We now describe the proposed technique used to construct a weighted graph of an image. One of the main drawbacks of any graph compression technique is that the graph itself has to be transmitted to the decoder. For this reason, the cost of transmitting the graph should be as small as possible. In our work, we pay particular attention to develop techniques that simplify as much as possible the graph structure without a significant decrease in the compression performance.

3.1.1 Graph weight metric

The graph structure used is a square grid where each pixel is a vertex of the graph and is connected to each of its 4-connected neighbors. We have chosen this structure because it has been proved that, when the graph is a 4-connected grid and all edges have the same weight, the 2D DCT basis functions are eigenvectors of the graph Laplacian, and thus the transform matrix U can be the 2D DCT matrix [50].

The edge weights of the graph represent the similarity between the two pixels connected by the edge. Several weighting functions have been used to determine the edge weights in image processing applications, two of the most commonly used are the Cauchy function and the Gaussian function [39], that are defined as follows:

$$\text{Cauchy function:} \qquad W(\mathcal{G})_{ij} = \frac{1}{1+(\frac{d_{ij}}{\alpha})^2},$$

$$\text{Gaussian function:} \qquad W(\mathcal{G})_{ij} = e^{-\frac{d_{ij}^2}{\sigma^2}},$$

where d_{ij} is the Euclidean distance between pixel i and j of the image $\mathbf{x}$ ($d_{ij} = |x_i - x_j|$) and α and σ are defined as in [51].

Most of the graph-based image compression methods present in the literature use an unweighted graph, even if in some case a weighted graph with Gaussian weights is used, as in [52]. Conversely, we propose to employ a Cauchy function

to determine the weights. In Section 3.2.2 we show that this choice outperforms the Gaussian weights.

Since the graph weights depend on the distance between pixels, the graph construction could be influenced by the presence of noise and the obtained graph may not capture the main characteristic of the image. In order to reduce this influence, we do not compute the edge weights using the original image, but first we smooth the image. It is important to use smoothing techniques that do not modify the edges present in the image, in our tests we used anisotropic diffusion [53].

3.1.2 Quantization

If we use a graph defined as in the previous section, the information we need to encode are only the edge weights, whereas the graph topology is fixed and there is no need to transmit it. Given an image block of n^2 pixels, its grid graph has $2n(n-1)$ edges, i.e. almost twice the number of pixels, which explains why it is extremely important to study techniques for reducing this overhead.

First, we decrease the number of possible edge weight values for each edge in the graph. To do this we quantize the argument of the weight function, i.e the distances d_{ij}. We have seen that their probability distribution can be approximated with a Laplacian; therefore, we use an uniform quantizer for Laplacian source [54] with an overload region.

Using this method, we can arbitrarily reduce the number of edge weight values down to two. In term of compression performance, the case with only two possible values is the most interesting because it has the best ratio between quality gain and cost for the graph transmission.

The graph obtained using only binary weight values is similar to an unweighted graph (i.e. with weights in $\{0,1\}$) such as the one used in [6, 7], with the important difference that, with the quantized weights, the graph cannot be disconnected. In fact, if the graph is disconnected, the graph transform does not perform well, in particular when there are connected components with a small number of nodes.

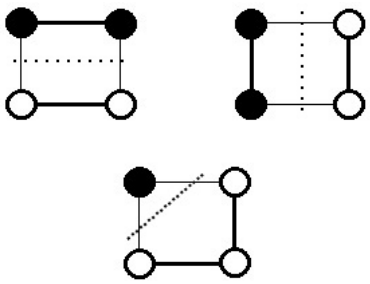


Fig. 3.1 Prediction-based graph construction method. The nodes represent the binary image transmitted to the decoder: the black ones are the edge pixels, the white ones the non-edge pixels. The dotted lines represent the predicted image discontinuity, the figure represents the three possible situations. The edges are set in accordance with the discontinuities: the ones with a thicker line have the higher weight, the ones with a thinner line have the lower weight.

In the following sections, we will focus on this binary case, developing an optimized graph compression scheme.

3.1.3 Graph edge prediction method

When we have only two possible edge weight values, the majority of the edges in the graph will typically have the higher edge weight, meaning that the two pixels connected by the edge are similar, instead only a small number of edges will have the lower edge weight, indicating that there is a discontinuity between the two pixels. Every interior node is connected to four pixels. In order to structure our prediction mechanism, we consider nodes in raster order and, for each node, we consider two edges, namely the one that connects the pixel to the nearest one in the next column, and the one that connects the pixel to the bottom one in the next row. We label each pixel of the image as "edge" or "non-edge" pixel. Specifically, a pixel is labeled as an edge pixel if at least one of the corresponding edges has the lower edge weight, otherwise it is labeled as a non-edge pixel.

In order to obtain a more compact representation of the graph structure, we have developed a method of edge prediction that reconstructs the graph edges starting from a binary image that specify if each pixel has an edge or non-edge label. Analysing the labels of neighboring pixels, the encoder can predict whether the discontinuity is horizontal, vertical or diagonal, as shown in

Algorithm 1 Prediction-based Graph Construction Algorithm. I_b: binary image, M: higher edge weight, m: lower edge weight

```
Set all edge weights equal to M;
for every edge pixel in I_b do
  N_hor= number of horizontal neighbors that are edge pixel;
  N_ver= number of vertical neighbors that are edge pixel;
  if N_hor > 0 then
    Set the vertical edge to m;
  end if
  if N_ver > 0 then
    Set the horizontal edge to m;
  end if
  if N_hor = 0 and N_ver = 0 then
    Set the vertical edge to m;
    Set the horizontal edge to m;
  end if
end for
```

Fig. 3.1. Then it constructs a new graph, setting the edge weights in accordance with the predicted discontinuities. The graph generation algorithm is explained in detail in Algorithm 1. The binary image containing the labels is losslessly coded and transmitted to the decoder, then, starting from the received labels, the decoder runs the graph construction algorithm and recovers the same graph used at the encoder. Several techniques could be used to code the binary image, in the proposed method we decided to use JBIG, however the use of other more specific techniques for contour coding, such as [55], will be evaluated in future.

As will be shown in Section 3.2.2, the graph provided by Algorithm 1 defines a graph Fourier transform with very high coding efficiency.

3.1.4 Deletion of isolated edges

In order to obtain a smoother binary image and to remove small discontinuities, we delete the connected components present in the binary image whose dimension is smaller than a threshold value, which was experimentally set to 10.

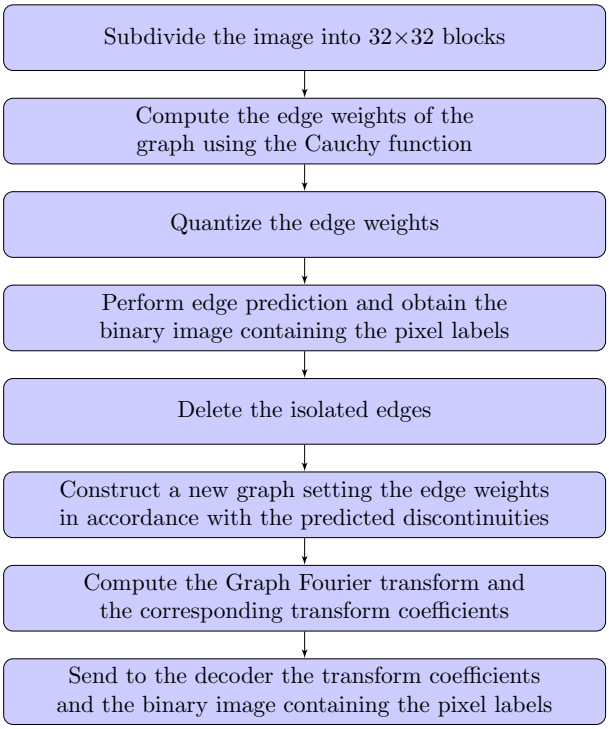


Fig. 3.2 Block diagram of the proposed method

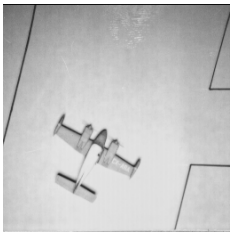
3.2 Experimental results

To test the proposed method, we have subdivided the images in 32×32 blocks and we constructed the graph of each image block with the techniques discussed in the previous section. After having obtained the graph, we computed the adjacency matrix $W(\mathcal{G})$ and the Laplacian matrix $L(\mathcal{G})$. We used as transform matrix the matrix Ψ of the eigenvectors of the Laplacian.Then, to code the transform coefficients we used a bit plane coding on each block and we estimated the bit rate computing the entropy of each bitplane. The principal steps of the proposed image compression method are summarized in Fig. 3.2.

We have applied our method to some standard images, three of which are shown in Fig. 3.3. We have chosen different image types, some having very sharp edges, such as airplane, while others are more natural, such as cameraman.

(a) Clock

(b) Cameraman



(c) Airplane

Fig. 3.3 Original images.

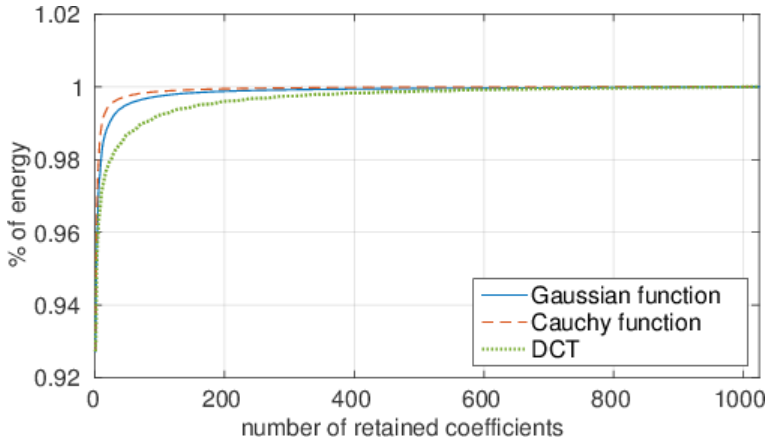


Fig. 3.4 Percentage of energy in function of the number of retained coefficients.

3.2.1 Edge weight metric evaluation

We have compared the performance of the two weighting functions showed in Section 3.1.1 by computing the percentage of signal energy in function of the number of retained coefficients. We have found that the Cauchy function has better compression performance than the Gaussian function, as shown in Fig. 3.4. For this reason, in our tests we used the Cauchy weighting function.

3.2.2 Graph compression

We have compared the performance of the graph transform using a graph with unquantized weights, with quantized weights and with the predicted weights. We computed the percentage of signal energy in function of the number of retained coefficients. The results are shown in Fig. 3.5. We can see that using a small number of edge weight values reduces the performance but not in a

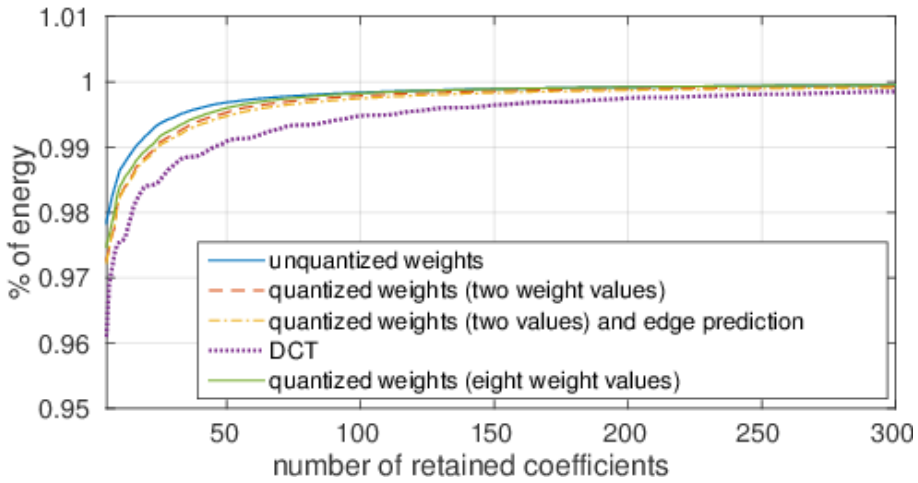


Fig. 3.5 Percentage of energy in function of the number of retained coefficients.

significant way. Moreover, it is important to note that the edge prediction method produces a graph that is a very good approximation of the original one and the results obtained are nearly the same, but it nearly halves the size of the graph overhead, resulting in large performance improvement.

3.2.3 Image compression performance

To evaluate the performance of the proposed Predictive Graph Transform (PGT), we computed the PSNR of each image as a function of the bitrate. We compared the proposed PGT with the standard DCT and with the edge-adaptive transform (EAT) proposed by Shen et al. in [6]. In order to have a fair comparison, we used the same block dimension and the same method for coding the transform coefficients. For the EAT and our proposed transform, the bitrate also takes into account the cost of transmitting the graph. Fig. 3.6 shows the results obtained. We can see that our method outperforms both the standard DCT and the EAT, with an average gain of approximately 1.6 dB over the DCT and a maximum gain of 3 dB. In Fig. 3.7, we show two examples of visual comparison between our proposed method and DCT where we can clearly see that the proposed PGT provides a significant quality gain.

In this chapter, we have not taken into account the use of intra prediction. In future, it would be interesting to investigate the possible application of the PGT to intra-prediction residuals.

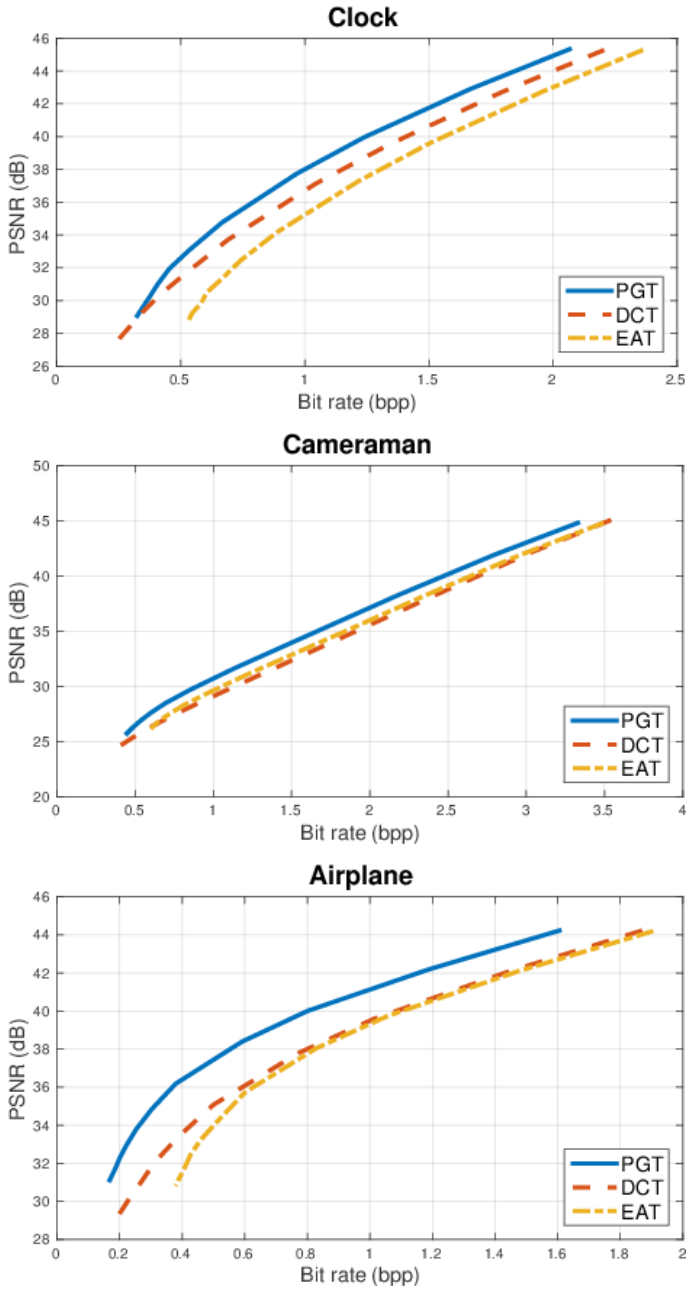
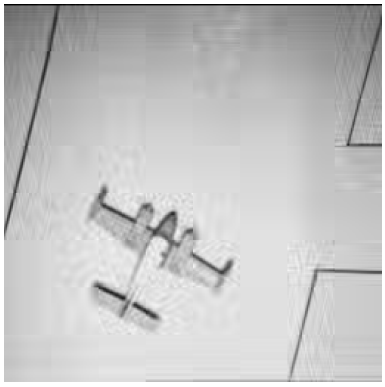
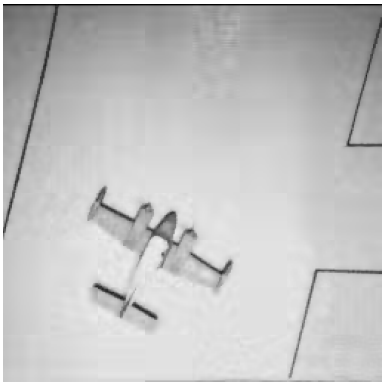


Fig. 3.6 RD curve comparison between our proposed method, DCT and EAT.



DCT



PGT

DCT

PGT

Fig. 3.7 Visual comparison between our proposed method and DCT at 0.4 bpp for the first line and 1.25 bpp for the second line.

Chapter 4

Graph Transform Learning for Image Compression

One of the biggest challenges in graph-based image compression is the choice of the graph and the corresponding transform. A good graph for effective transform coding should lead to easily compressible signal coefficients, at the cost of a small overhead for coding the graph. Since the definition of the graph is often not straightforward, the problem of designing a graph transform stays critical and may actually represent the major obstacle towards effective compression of images.

In this chapter, we propose a novel graph-based framework for effective coding of images that takes into account the coding of the images as well as the cost of transmitting the graph. In particular, we introduce an innovative way for coding the graph by treating its edge weights as a graph signal that lies on the dual graph. We then compute the graph Fourier transform of this signal and code its quantized transform coefficients. The choice of the graph is posed as a rate-distortion optimization problem that is cast as a graph learning problem. The cost of coding the image signal is captured by minimizing the smoothness of the image on the learned graph while the transmission cost of the topology is controlled by penalizing the sparsity of the graph Fourier coefficients of the edge weight signal that lies on the dual graph. The solution of

Part of the work described in this chapter has been previously published in G. Fracastoro, D. Thanou, P. Frossard, Graph Transform Learning for Image Compression, *Proc. of Picture Coding Symposium*, 2016.

our optimization problem is a graph that provides an effective tradeoff between the quality of the transform and its transmission cost. Experimental results on natural images confirm that the proposed algorithm can efficiently infer meaningful graph topologies, which eventually lead to improved coding results compared to non-adaptive methods based on DCT.

A few attempts have been recently proposed to learn the structure and in particular a graph from data observations. In [56], the authors formulate the graph learning problem as a precision matrix estimation with generalized Laplacian constraints. In [57], a sparse combinatorial Laplacian matrix is estimated from the data samples under a smoothness prior. In [58], a new class of transforms called graph template transform is proposed; the authors use a graph template to impose a sparsity pattern and approximate the empirical inverse covariance based on that template. Even if all these methods contain some constraints on the sparsity of the graph, none of them takes into account the real cost of representing, and thus coding, the graph. Instead, in this work, we go beyond prior art and we fill this gap by defining a new graph learning problem for image compression that takes into account this overhead.

4.1 Basic definitions on graphs

For any weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the graph Laplacian $L(\mathcal{G}) = D(\mathcal{G}) - W(\mathcal{G})$ can also be defined using the incidence matrix $B(\mathcal{G}) \in \mathbb{R}^{N \times M}$ [59] such that

$$B(\mathcal{G})_{ie} = \begin{cases} 1, & \text{if } e = (i,j) \\ -1, & \text{if } e = (j,i) \\ 0, & \text{otherwise,} \end{cases}$$

where an orientation is chosen arbitrarily for each edge. Then, let $\widehat{W}(\mathcal{G}) \in \mathbb{R}^{M \times M}$ be a diagonal matrix where $\widehat{W}(\mathcal{G})_{ee} = W(\mathcal{G})_{ij}$ if $e = (i,j)$. We can define the graph Laplacian $L(\mathcal{G})$ as

$$L(\mathcal{G}) = B(\mathcal{G})\widehat{W}(\mathcal{G})B(\mathcal{G})^T. \tag{4.1}$$

It is important to underline that the graph Laplacian obtained using (4.1) is independent from the edge orientation in $\mathcal{G}$.

As shown in Chapter 2, given any graph signal $\mathbf{x} \in \mathbb{R}^N$ defined on the nodes of the graph $\mathcal{G}$, the smoothness of $\mathbf{x}$ on $\mathcal{G}$ can be measured using the Laplacian $L(\mathcal{G})$ [36]

$$\mathbf{x}^T L(\mathcal{G})\mathbf{x} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(x_i - x_j)^2. \tag{4.2}$$

Eq. (4.2) shows that a graph signal $\mathbf{x}$ is considered to be smooth if strongly connected nodes have similar signal values.

4.2 Graph-transform optimization

4.2.1 Rate-distortion tradeoff

Graph-based image compression methods use a graph representation of the image signal through its GFT, in order to obtain a data-adaptive transform which captures the main characteristics of the image. The GFT coefficients are then encoded, instead of the signal values. In general, a signal that is smooth on a graph has its energy concentrated in the low frequency coefficients of the GFT, hence it is easily compressible. To obtain good compression performance, the graph should therefore be chosen such that it leads to a smooth representation of the signal. On the other hand, it should also be easy to encode, since it has to be transmitted to the decoder for signal reconstruction. Often, the cost of the graph representation outweighs the benefits of using an adaptive transform for signal representation. In order to find a good balance between graph signal representation benefits and coding costs, we introduce a new graph learning approach that takes into consideration the above mentioned criteria.

We first pose the problem of finding the optimal graph as a rate-distortion optimization problem defined as

$$\min_{L(\mathcal{G})\in\mathbb{R}^{N\times N}} \mathcal{D}(L(\mathcal{G})) + \gamma(\mathcal{R}_c(L(\mathcal{G})) + \mathcal{R}_G(L(\mathcal{G}))), \tag{4.3}$$

where $\mathcal{D}(L(\mathcal{G}))$ is the distortion between the original image and the reconstructed one. The total coding rate is composed of two representation costs,

namely the cost of the transform coefficients $\mathcal{R}_c(L(\mathcal{G}))$ and the cost of the graph description $\mathcal{R}_G(L(\mathcal{G}))$. Each of these terms possibly depends on $L(\mathcal{G})$ and on the coding scheme. We describe them in more details in the rest of the section.

4.2.2 Distortion approximation

The distortion $\mathcal{D}(L(\mathcal{G}))$ is defined as follows

$$\mathcal{D}(L(\mathcal{G})) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_q\|^2,$$

where $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are respectively the original and the reconstructed image, and $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}_q$ are respectively the transform coefficients and the quantized transform coefficients. The equality holds due to the orthonormality of the GFT. Considering a uniform scalar quantizer with the same step size q for all the transform coefficients, if q is small the expected value of the distortion $\mathcal{D}(L(\mathcal{G}))$ can be approximated as follows [60]

$$\mathcal{D}(L(\mathcal{G})) = \frac{q^2 N}{12}.$$

With this approximation, the distortion depends only on the quantization step size and it does not depend on the chosen $L(\mathcal{G})$ [8]. For simplicity, in the rest of the paper we adopt this assumption. Therefore, the optimization problem (4.3) is reduced to minimizing the rate terms.

4.2.3 Rate approximation of the transform coefficients

We can evaluate the cost of the transform coefficients $R_c(L(\mathcal{G}))$ by using the approximation proposed in [8], [7]

$$\begin{aligned} \mathcal{R}_c(L(\mathcal{G})) &= \mathbf{x}^T L(\mathcal{G}) \mathbf{x} = \mathbf{x}^T \left(\sum_{l=0}^{N-1} \lambda_l \psi_l \psi_l^T \right) \mathbf{x} \\ &= \sum_{l=0}^{N-1} \lambda_l (\mathbf{x}^T \psi_l)(\psi_l^T \mathbf{x}) = \sum_{l=0}^{N-1} \lambda_l \hat{\mathbf{x}}_l^2, \end{aligned} \tag{4.4}$$

where λ_l and ψ_l are respectively the l-th eigenvalue and eigenvector of $L(\mathcal{G})$. Therefore, $\mathcal{R}_c(L(\mathcal{G}))$ is an eigenvalue-weighted sum of squared transform coefficients. It assumes that the coding rate decreases when the smoothness of the signal x over the graph defined by $L(\mathcal{G})$ increases. In addition, (4.4) relates the measure of the signal smoothness with the sparsity of the transform coefficients. The approximation in (4.4) does not take into account the coefficients that corresponds to $\lambda_0 = 0$ (i.e., the DC coefficients). Thus, (4.4) does not capture the variable cost of DC coefficients in cases where the graph contains a variable number of connected components. However, we impose that the graph is connected in our work, which removes the influence of the number of connected components. In this case, the cost of the DC coefficient is independent of $L(\mathcal{G})$, so we can avoid to consider it in the $\mathcal{R}_c(L(\mathcal{G}))$ approximation.

4.2.4 Rate approximation of the graph description

The graph description cost $\mathcal{R}_G(L(\mathcal{G}))$ depends on the chosen method to code the graph. In order to reduce the graph transmission cost, we choose to vary only the edge weights, without changing the connections between the nodes, which are described by the incidence matrix $B(\mathcal{G})$. Therefore, the graph can be defined only by a vector $\mathbf{w} \in \mathbb{R}^M$, where $\mathbf{w}_e$ is the weight of the edge e. Then, by using (4.1) we can define the graph Laplacian $L(\mathcal{G}) = B(\mathcal{G})^T \text{diag}(\mathbf{w}) B(\mathcal{G})$.

In order to compress the edge weight vector w, we propose to treat it as a graph signal that lies on the dual graph $\mathcal{G}_d$. Given a graph $\mathcal{G}$, its dual graph $\mathcal{G}_d$ is an unweighted graph where each node of $\mathcal{G}_d$ represents an edge of $\mathcal{G}$ and two nodes of $\mathcal{G}_d$ are connected if and only if their corresponding edges in $\mathcal{G}$ share a common endpoint. An example of a dual graph is shown in Fig. 4.1. We choose to use this graph representation for the signal w because consecutive edges usually have similar weights, so the dual graph can provide a smooth representation of w. Also for $\mathcal{G}_d$ we can define its graph Laplacian matrix $L(\mathcal{G}_d) \in \mathbb{R}^{M \times M}$ and the corresponding eigenvector and eigenvalue matrices $\Psi_d \in \mathbb{R}^{M \times M}$ and $\Lambda_d \in \mathbb{R}^{M \times M}$ such that $L(\mathcal{G}_d) = \Psi_d \Lambda_d \Psi_d^T$.

In the literature, the dual graph has already been used in graph learning problems. In [61] the authors propose a method for joint denoising and contrast enhancement of images using the graph Laplacian operator, where the weights

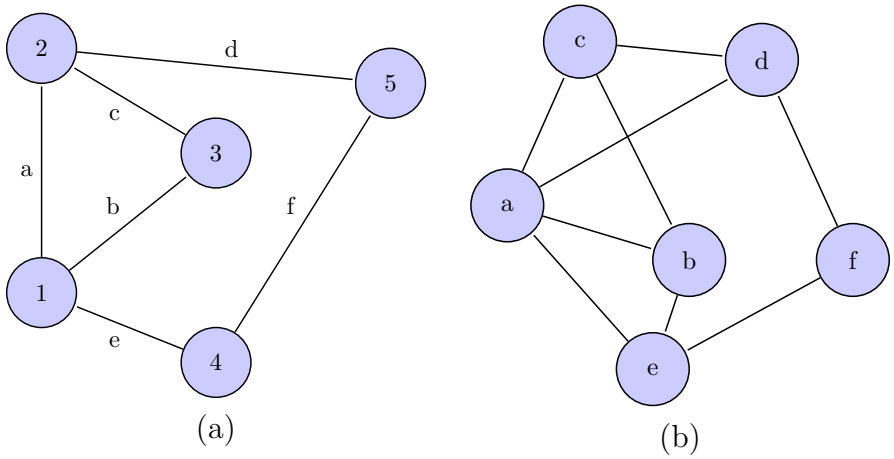


Fig. 4.1 An example of a graph (a) and its corresponding dual graph (b). The edges in the first graph (labeled with lower case letters) become the nodes of the corresponding dual graph.

of the graph are defined through an optimization problem that involves the dual graph. Moreover, [49] presents a graph-based dequantization method by jointly optimizing the desired graph-signal and the similarity graph, where the weights of the graph are treated as another graph signal defined on the dual graph.

Since $\mathbf{w}$ can be represented as a graph signal, we can compute its GFT $\hat{\mathbf{w}} \in \mathbb{R}^M$ as

$$\hat{\mathbf{w}} = \Psi_d^T \mathbf{w}.$$

Therefore, we can use $\hat{\mathbf{w}}$ to describe the graph and we evaluate the cost of the graph description by measuring the coding cost of $\hat{\mathbf{w}}$. It has been shown that the total bit budget needed to code a vector is proportional to the number of non-zero coefficients [62], thus we approximate the cost of the graph description by measuring the sparsity of $\hat{\mathbf{w}}$ as follows

$$\mathcal{R}_G(L(\mathcal{G})) = \|\hat{\mathbf{w}}\|_1 = \|\Psi_d^T \mathbf{w}\|_1. \tag{4.5}$$

The approximation of $\mathcal{R}_G(L(\mathcal{G}))$ presented in (4.5) may look similar to the one used in [49]. The main difference between the two formulations is that in (4.5) $\mathbf{w}$ is represented using the GFT, instead in [49] the authors use a difference operator to represent it.

4.2.5 Graph learning problem

By using (4.1), (4.4) and (4.5), the graph learning problem (4.3) becomes equivalent to the following optimization problem

$$\min_{w\in\mathbb{R}^M} \mathbf{x}^T B(\mathcal{G})(\mathrm{diag}(\mathbf{w}))B(\mathcal{G})^T\mathbf{x} + \alpha\|\Phi^T\mathbf{w}\|_1, \tag{4.6}$$

where α is a constant parameter.

Building on the rate-distortion formulation of (4.6), we find the optimal graph topology by solving the following optimization problem

$$\begin{aligned} &\min_{w\in\mathbb{R}^M} \mathbf{x}^T B(\mathcal{G})(\mathrm{diag}(\mathbf{w}))B(\mathcal{G})^T\mathbf{x} + \alpha\|\Phi^T\mathbf{w}\|_1 - \beta\mathbf{1}^T\log(\mathbf{w}), \\ &\text{s. t.} \quad \mathbf{w} \leq \mathbf{1}, \end{aligned} \tag{4.7}$$

where α and β are two positive regularization parameters and $\mathbf{1}$ denotes the constant one vector. The inequality constraint has been added only to guarantee that all the weights are in the range $(0, 1]$, which is the same range of the most commonly used weighting functions [39]. Instead, the logarithmic term has been added to penalize low weight values and to avoid the trivial solution. In addition, this term guarantees that $w_m > 0$, $\forall m$, so that the graph is always connected. A logarithmic barrier is often employed in graph learning problems [63] and it has been shown that a graph with Gaussian weights can be seen as the result of a graph learning problem with a specific logarithmic barrier on the edge weights [63].

The problem in (4.7) can be cast as a convex optimization problem with a unique minimizer. To solve problem (4.7), we write the first term in the following form

$$\begin{aligned} \mathbf{x}^T B(\mathcal{G})(\mathrm{diag}(\mathbf{w}))B(\mathcal{G})^T\mathbf{x} &= \mathrm{tr}((B(\mathcal{G})^T\mathbf{x}\mathbf{x}^T B(\mathcal{G}))\mathrm{diag}(\mathbf{w})) \\ &= \mathrm{vec}(B(\mathcal{G})^T\mathbf{x}\mathbf{x}^T B(\mathcal{G}))^T\mathrm{vec}(\mathrm{diag}(\mathbf{w})) \\ &= \mathrm{vec}(B(\mathcal{G})^T\mathbf{x}\mathbf{x}^T B(\mathcal{G}))^T M_{\mathrm{diag}}\mathbf{w}, \end{aligned}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix, $\mathrm{vec}(\cdot)$ is the vectorization operator, and $M_{\mathrm{diag}} \in \mathbb{R}^{M^2\times M}$ is a matrix that converts the vector w in $\mathrm{vec}(\mathrm{diag}(w))$.

Then, we can rewrite problem (4.7) as

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^M} & \operatorname{vec}(B(\mathcal{G})^T\mathbf{x}\mathbf{x}^T B(\mathcal{G}))^T M_{\text{diag}}\mathbf{w} + \alpha\|\Psi_d^T\mathbf{w}\|_1 - \beta\mathbf{1}^T\log(\mathbf{w}), \\ \text{s. t.} & \quad \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{4.8}$$

The problem in (4.8) is a convex problem with respect to the variable w and can be solved efficiently via interior-point methods [64].

4.3 Image compression application

We now describe how the above graph learning problem can be applied to image compression. As pointed out in the previous sections, we have two different information to transmit to the decoder: the transform coefficients of the image signal $\hat{\mathbf{x}}$ and the description of the graph $\hat{\mathbf{w}}$. The transform coefficients are quantized using a uniform quantizer with the same step size q for all the coefficients. Then, we code the quantized coefficients until the last non-zero coefficient using an adaptive bitplane arithmetic encoder and we transmit the position of the last significant coefficient.

To code the graph, we use its GFT coefficients vector $\hat{\mathbf{w}}$. In order to reduce the cost of the graph description, we reduce the number of elements in $\hat{\mathbf{w}}$ taking into account only the first $\widetilde{M} \ll M$ coefficients, which usually are the most significant, and setting the other $M - \widetilde{M}$ coefficients to zero. The reduced vector $\hat{\mathbf{w}}_r \in \mathbb{R}^{\widetilde{M}}$ is then quantized and coded with the same entropy coder used for the image signal.

The principal steps of the proposed image compression method are summarized in Fig. 4.2. Given an image signal, we first solve the optimization problem in (4.8) obtaining the optimal solution $\mathbf{w}^*$. To transmit $\mathbf{w}^*$ to the decoder, we first compute its GFT coefficients $\hat{\mathbf{w}}^*$ and the reduced vector $\hat{\mathbf{w}}_r^*$, then we quantize and code it using an entropy coder. The reconstructed graph described by $\tilde{\mathbf{w}}^*$ is then used to define the GFT transform for the image signal.

It is important to underline that, since we perform a quantization of $\hat{\mathbf{w}}_r^*$, the reconstructed signal $\tilde{\mathbf{w}}^*$ is not equal to the original $\mathbf{w}^*$ and its quality

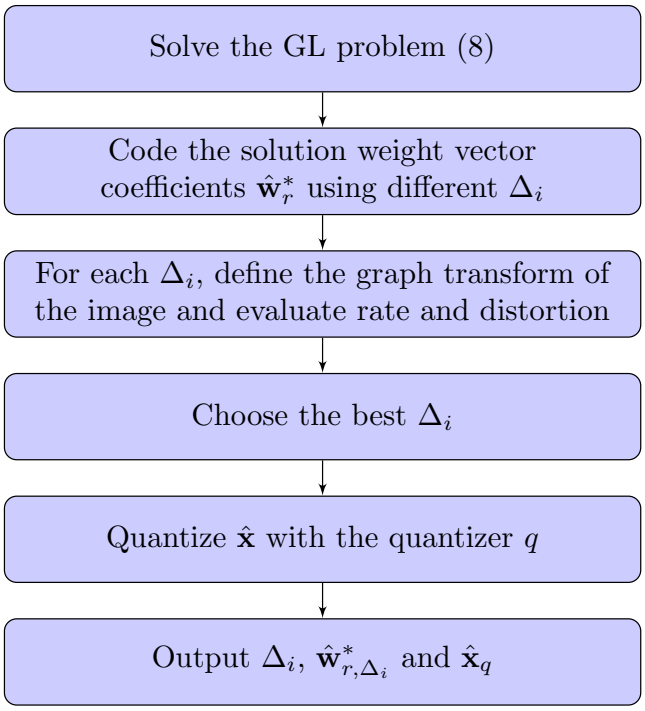


Fig. 4.2 Block diagram of the proposed method.

depends on the quantization step size used. Since it is important to find the best tradeoff between the quality of the graph and its transmission cost, for each block we test different quantization step sizes $\{\Delta_i\}_{1\leq i\leq Q}$ for a given graph represented by $\hat{\mathbf{w}}_r^*$. To choose the best quantization step size, we use the following rate-distortion problem

$$\min_i \mathcal{D}(\Delta_i) + \gamma(\mathcal{R}_c(\Delta_i) + \mathcal{R}_G(\Delta_i)), \tag{4.9}$$

where $\mathcal{R}_G(\Delta_i)$ is the rate of $\hat{\mathbf{w}}_{r,\Delta_i}^*$ (the coefficient vector $\hat{\mathbf{w}}_r^*$ quantized with Δ_i), $\mathcal{D}(\Delta_i)$ and $\mathcal{R}_c(\Delta_i)$ are respectively the distortion and the rate of the reconstructed image signal obtained using the graph transform described by $\hat{\mathbf{w}}_{r,\Delta_i}^*$. We underline that in (4.9) we evaluate the actual distortion and rate without using the approximation introduced previously in (4.3), (4.4), (4.5). The coding methods described previously are used to compute the rates $\mathcal{R}_c(\Delta_i)$ and $\mathcal{R}_G(\Delta_i)$.

4.4 Experimental results

In this section, we evaluate the performance of the proposed method. We first describe the general experimental setting, then we present the experimental results obtained.

4.4.1 Experimental setup

We test our method on four standard grayscale images (Lena, Boat, Peppers and House) and we split them into non-overlapping 16×16 pixel blocks. The chosen topology of the graph is a 4-connected grid: this is the most used graph topology for graph-based image compression, since its number of edges is not too high, and thus the coding cost is limited. In a 4-connected square grid with N nodes, we have $M = 2\sqrt{N}(\sqrt{N}-1)$ edges. In all our experiments, we set $\widetilde{M} = 64$ and $Q = 8$. To find the best value for the parameters α and β of the graph learning problem in (4.8), we use the following strategy. The value of the parameter α depends on the characteristics of the block. For this reason, we perform a block classification using the structure tensor analysis, as done in [65]. Let μ_1 and μ_2 be the two eigenvalues of the structure tensor, where $\mu_1 \geq \mu_2 \geq 0$, we can subdivide the image blocks in the following way:

- Class 1: smooth blocks, if $\mu_1 \approx \mu_2 \approx 0$;
- Class 2: blocks with a dominant principal gradient, if $\mu_1 \gg \mu_2 \approx 0$;
- Class 3: blocks with a more complex structure, if μ_1 and μ_2 are both large.

Fig. 4.3 shows an example of block classification. We set $\alpha = 100$ for blocks that belong to the first class, $\alpha = 500$ for blocks that belong to the second class and $\alpha = 800$ for blocks that belong to the third class. For all the three classes, we set $\beta = 1$ in the graph learning problem.

We compare the performance of the proposed method against the classical DCT transform. To have a fair comparison, we code the transform coefficients $\hat{\mathbf{x}}$ of the image signal using the same entropy coder for the graph-based method and for DCT-based encoder. In the first case, in addition to the bitrate of

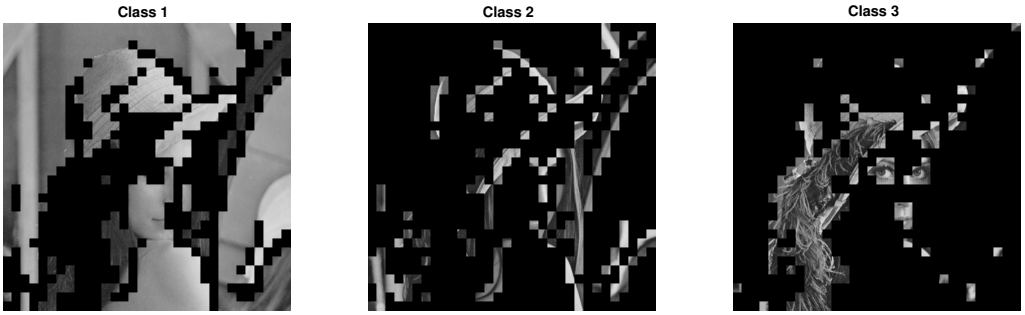


Fig. 4.3 Block classification of Lena.

$\hat{\mathbf{x}}$, we count the bitrate due to the trasmission of $\hat{\mathbf{w}}^*_{r,\Delta_i}$ and 3 additional bits per block to transmit the chosen quantization step size Δ_i for $\hat{\mathbf{w}}_r$. For both methods, we vary the quantization step size q of the transform coefficients to vary the encoding rates.

Finally, in our method for each block we compare the RD-cost of the GFT and the one of the DCT. Then, we code the block with the transform that has the lowest RD-cost and we use 1 additional bit per block to signal if we are using the GFT or the DCT.

Image	class 1	class 2	class 3
Lena	0.06	0.66	0.52
Boat	0.03	0.31	0.47
Peppers	0.10	0.77	0.89
House	0.02	0.63	0.62

Table 4.1 Average gain in PSNR measured with the Bjontegaard metric.

4.4.2 Results

In Fig. 4.4, we show the performance of the two methods on the image House. More results are given in Table 4.1, where we use the Bjontegaard metric [66] to compute the average gain in PSNR compared to the DCT. In the second and third classes, the proposed method outperforms DCT providing an average PSNR gain of 0.6 dB for blocks in the second class and 0.64 dB for blocks in the third class. It is interesting to point out that there is not a significant

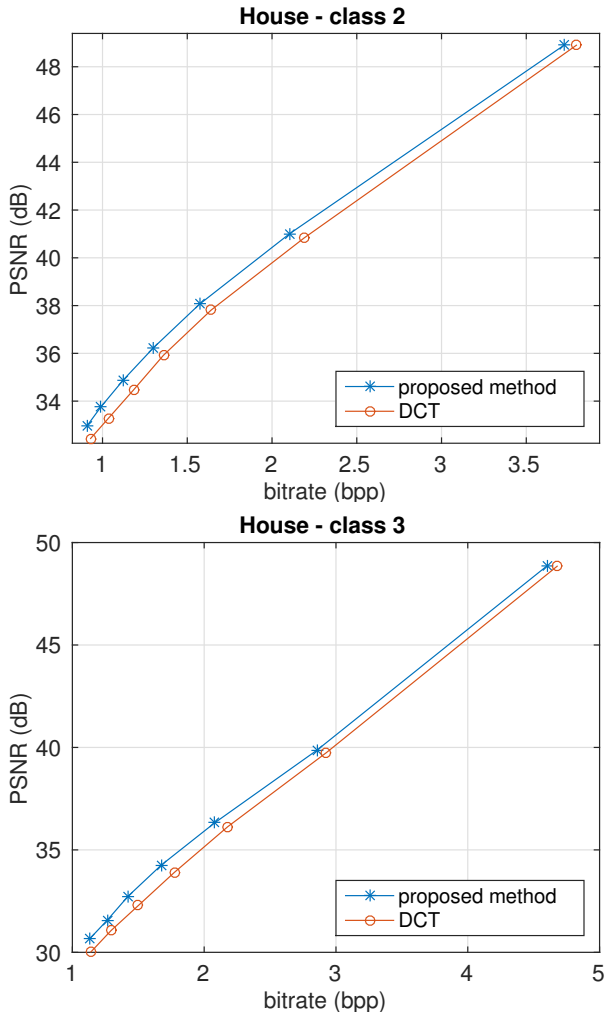


Fig. 4.4 RD comparison between the proposed method and DCT.

difference in performance between the second class and the third one. Instead, in the first class the gain is nearly 0, as DCT in this case is already optimal.

The obtained results show that the proposed method can outperform classical fixed transforms as DCT, even if they could be further improved by optimizing the coding method, in particular the quantization strategy.

We point out that in this chapter we do not take into account intra-prediction. In future, it would be interesting to investigate a possible application of the proposed method to intra-prediction residuals and define a specific graph learning problem also for this type of signals.

Chapter 5

Superpixel-driven graph transform for image compression

In this chapter, we propose a novel graph transform approach aiming at reducing the cost of transmitting the graph structure while retaining the advantage of a shape-adaptive and edge-aware operator. To this end, the image is first segmented into uniform regions that adhere well to image boundaries. Such a goal can be achieved using the so-called superpixels, which are perceptually meaningful atomic regions which aim at replacing rigid pixel grid. Examples of algorithms used to generate these kind of regions are Turbopixel [67], VCells [68] and the widely used and very fast SLIC algorithm [69]. Then, we propose to apply a graph transform within each superpixel that, being homogeneous region, can be efficiently represented using an uniform graph, i.e. all graph edges are given the same weight. In this way, the overhead of representing the graph structure within each superpixel is avoided. Nonetheless, we need to transmit additional information to describe region boundaries. To limit such coding overhead, we design a clustering method that is able to aggregate superpixels, thus reducing the number of regions that need to be coded.

Part of the work described in this chapter has been previously published in G. Fracastoro, F. Verdoja, M. Grangetto, E. Magli, Superpixel-driven Graph Transform for Image Compression, *Proc. of IEEE International Conference on Image Processing*, 2015, pp. 2631-2635.

The use of superpixels in compression is still an almost unexplored research field and up to date only few works investigated the topic. Moreover, the proposed approaches work in very specific cases, e.g. texture compression [70] or user-driven compression [71]. On the contrary, the joint exploitation of graph transforms and superpixels as a general approach to image compression is completely novel and represents the key idea in this work. The contributions of this work are the definition of a superpixel-driven graph transform, its rate/distortion analysis using a bitplane encoding approach and the comparison with standard DCT transform.

5.1 Proposed technique

Given an image $\mathbf{x} \in \mathbb{R}^N$ of N pixels, the proposed Superpixel-driven Graph Transform (SDGT) performs the following steps:

- divide $\mathbf{x}$ in m regions by using SLIC [69], an example of superpixel segmentation is shown in Fig. 5.1;
- cluster similar superpixels, to reduce the number of borders to be coded to a desired number m';
- inside each region, compute a piecewise smooth graph transform.

Superpixels are used to get a computationally efficient segmentation of the image into homogeneous regions, that can be modeled with simple uniform graph structure for the following transform stage.

5.1.1 Superpixel clustering

In this section the preliminary segmentation step based on superpixel is described.

We define an m-regions segmentation of an image $\mathbf{x}$ as a partition $\mathcal{P}^m = \{l_i\}_{i=1}^m$ of the pixels in $\mathbf{x}$; more precisely:

$$\begin{aligned} &\forall x_i \text{ where } 1 \leq i \leq N, \exists l \in \mathcal{P}^m \mid x_i \in l \\ &\forall l \in \mathcal{P}^m, \nexists\, l' \in \mathcal{P}^m - \{l\} \mid l \cap l' \neq \emptyset \end{aligned} \tag{5.1}$$

Fig. 5.1 An image segment into superpixels.

Starting from an image $\mathbf{x}$ and a partition $\mathcal{P}^m$ composed of m regions, output by some superpixel algorithm, the proposed algorithm aims at merging at each iteration the pair of labels representing the most similar regions between the ones determined in the previous step until the desired number of regions $m' < m$ is reached. In particular at the kth iteration the two most similar segments of $\mathcal{P}^k$ are merged to obtain a new set $\mathcal{P}^{k-1}$ composed of $k-1$ segments. This process can be iterated for $k = m, m-1, \ldots, m'$, generating a hierarchy of regions in terms of their respective similarity. The number of regions m' to be clustered must be chosen as a tradeoff between the segmentation accuracy and the coding overhead required to represent and compress the borders of the regions as discussed in more detail in Section 5.2.

We represent the merging process using a weighted graph. An initial undirected weighted graph $\mathcal{G}^m = (\mathcal{P}^m, \mathcal{E}^m)$ is constructed over the superpixel set $\mathcal{P}^m$, and the set $\mathcal{E}^m$ is the edge set defined in the following way

$$\mathcal{E}^m = \{w_{ij}^m,\ \forall i \neq j \mid l_i^m, l_j^m \in \mathcal{P}^m \wedge C(l_i^m, l_j^m) = 1\} \tag{5.2}$$

for some region adjacency function C, i.e. 4-connectivity. Since G^m is an undirected graph we have that $w_{ij}^m = w_{ji}^m$; the weights represent the distance (or dissimilarity measure) between a pair of regions $w_{ij}^m = d(l_i^m, l_j^m)$.

The approach proposed here can be used in conjunction with several distance metrics capable to capture the dissimilarity between a pair of segmented regions. In this study, CIELAB color space and the standard CIEDE2000 color difference [72] have been chosen thanks to their ability to reliably cluster similar superpixels as shown in [73]. Given two regions l_i and l_j, we compute the mean values of the L*a*b* components $M_i = (\mu_{L^*,i}, \mu_{a^*,i}, \mu_{b^*,i})$ and $M_j = (\mu_{L^*,j}, \mu_{a^*,j}, \mu_{b^*,j})$, and we define the distance between the two labels as

$$d(l_i, l_j) = \Delta E_{00}(M_i, M_j) \tag{5.3}$$

where ΔE_{00} is the CIEDE2000 color difference [72].

At each iteration k, we pick the pair of labels $l_p^k, l_q^k \in P^k$ having

$$w_{pq}^k = \min_{w_{ij}^k \in \mathcal{E}^k} \{w_{ij}^k\}$$

and merge them; as a consequence a new partition $\mathcal{P}^{k-1} = \mathcal{P}^k - \{l_q^k\}$ is formed. The new partition $\mathcal{P}^{k-1}$ now comprises $k-1$ segments and all the pixels $x_i \in l_p^k \cup l_q^k$ are assigned to the label l_p^{k-1}. After that, edges and corresponding weights needs to be updated as well. The set $\mathcal{E}^{k-1}$ is generated according to the following rule:

$$w_{ij}^{k-1} = \begin{cases} d(l_p^{k-1}, l_j^{k-1}) & \text{if } i = p \vee i = q \\ w_{ij}^k & \text{otherwise} \end{cases} \tag{5.4}$$

It must be noted that w_{pq}^k is no longer included in $\mathcal{E}^{k-1}$ since the corresponding label has been merged into a single one.

When $k = m'$, the algorithm stops returning the partition $\mathcal{P}^{m'}$ composed of the desired number of regions. A segmentation example with $m' = 100$ is shown in Fig. 5.2.

Fig. 5.2 An image divided into 100 regions by the proposed algorithm.

5.1.2 Intra-region graph transform

Now we move to the description of the graph transform employed within each region that leads to the computation of the proposed SDGT.

Given a m'-regions segmentation $P^{m'}$ of the image I, in each segment $\mathbf{x}$ of $\mathcal{P}^{m'}$ we can define an unweighted graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$, where the nodes are the pixels of the segment l and $\mathcal{E}_l$ is the set of edges. The adjacency matrix $A(\mathcal{G}_l)$ is defined in the following way:

$$A(\mathcal{G}_l)_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{N}_l \wedge i, j \in l \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

where $\mathcal{N}_i$ is the set of 4-connected neighbors of the pixel i.

The adjacency matrix is used to compute the Laplacian matrix $L(\mathcal{G}_l)$. Then, the matrix Ψ whose columns are the eigenvectors of $L(\mathcal{G}_l)$ is used to compute the graph Fourier transform (GFT).

It is important to underline that to construct the graph we only need the information about the coordinates of the region borders, that can be easily summarized in a binary image. In this way, the cost for transmitting the graph structure is considerably reduced and the GFT is used as an effective transform

for the arbitrarily shaped regions computed by the algorithm described in Section 5.1.1. Finally, we refer to the whole set of transformed regions as the SDGT of the entire image.

5.2 Experimental results

To evaluate the performance of the proposed SDGT, we need to take into account its energy compaction ability and the cost for coding overhead information, i.e. the region-borders.

A popular and simple method for evaluating the transform compaction efficiency is to study the quality of the reconstructed image, e.g. using PSNR with respect to the original image, as a function of the percentage of retained transformed coefficients [74]; albeit interesting, this approach would neglect the cost required to encode the ancillary information required to compute the inverse transform.

To overcome this problem, in the following we estimate the coding efficiency provided by SDGT by considering bit plane encoding of SDGT transformed coefficients. Each bitplane is progressively extracted, from the most significant down to the least significant one, and the bitrate of each bitplane is estimated by its entropy. To this end, each bitplane is modeled as an independent and memoryless binary source.

It is worth pointing out that such an estimate represents an upper bound to the actual bitrate that would be obtained using a proper entropy coding algorithm that is likely to exploit further the residual spatial correlation of the transformed coefficients and the dependency between different bitplanes. Nonetheless, the proposed bitplane approach can be replicated on any other transform, e.g. the standard 8×8 DCT, allowing us to analyze the achievable gain in a fair way.

Finally, to estimate the SDGT penalty due to coding of the region borders, we use the standard compression algorithm for bi-level images JBIG [75]. The regions boundaries are represented as a binary mask that is then compressed with JBIG, whose bitrate is considered as coding overhead; from our experimentation we have seen that this overhead is, on average, around 0.06 bpp. The

use of other more specific methods for transmitting the region borders, such as chain code [76, 77] and arithmetic edge coding [55], will be evaluated in future. These methods are designed for efficiently encoding region borders. However, the context of segmentation borders coding presents one characteristic that these methods are not tailored to [78]: since all pixels must be assigned to a region, all borders are shared between the two regions. It follows that, with these methods, all edges will be encoded twice, resulting in an inefficient coding method. Therefore, applying these methods will require careful investigation in order to tailor them to the specific requirements of our technique.

Therefore using bitplane coding and JBIG we get a rough estimation of the total bitrate needed to code the image with the SDGT transform. We compare the obtained results with the standard DCT computed on 8×8 blocks. As proved by Zhang and Florêncio in [50], if the graph is a uniform 4-connected grid the 2D DCT basis functions are eigenvectors of the graph Laplacian, and thus the graph Fourier transform matrix Ψ turns to be the 2D-DCT matrix. Therefore, the 8×8 DCT can be seen as a graph transform like the SDGT, with the major difference that instead of using superpixels as coding blocks it uses a fixed grid of 8×8 blocks.

We have tested the transforms on several images from a dataset of lossless images widely used in compression evaluation [79]. All the images in that dataset are either 768×512 or 512×768 in size.

In Fig. 5.3 three sample images are shown and the respective coding results (PSNR in dB vs. bitrate measured in bit per pixel) are shown in Fig. 5.4; these results have been obtained setting $m = 600$, $m' = 100$ and coding the luminance component only.

We can see that SDGT significantly outperforms the DCT, in particular at low bitrate, where it is able to achieve a maximum gain of more than 2 dB. Overall, the average gain obtained is approximately 1 dB. This achievement is particularly significant if one recall that the SDGT bitrate includes the constant penalty yielded by JBIG coding of the borders. A detail of the significant improvement at low bitrate obtained by SDGT can be visually appreciated in Fig. 5.5.

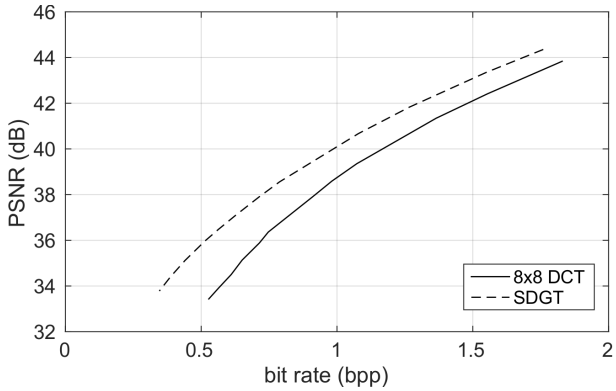
Since standard image compression data set are historically biased by low resolution images we conclude our analysis by considering high resolution

(a) Parrots (b) Lighthouse (c) Girl

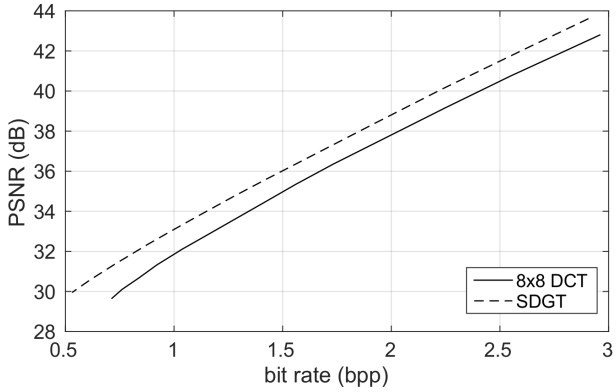
Fig. 5.3 Three of the sample images.

images that are typically acquired by current imaging devices. We have tested our method and the 8×8 DCT on some HD images acquired using a DSLR camera; in particular, for complexity reasons, we have applied SDGT to non trivial 512×512 patches cropped from the original images. In Fig. 5.6 the results obtained on a sample image are shown; it is worth pointing out that the SDGT gain over DCT is larger in this case and span all the considered bitrate range. This is due to the fact that regions in HD images are usually wider and smoother and therefore the segmentation algorithm and, consequently, the graph transform can be even more effective.

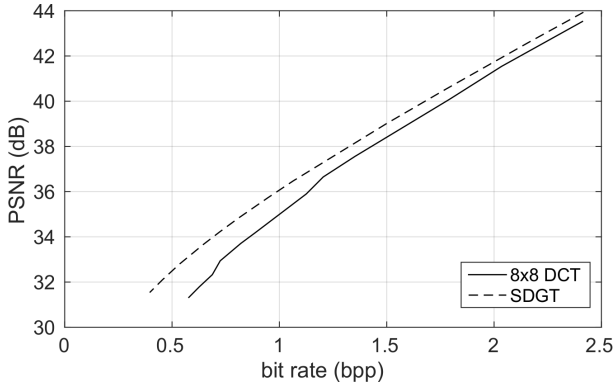
Finally, it is important to underline that in this chapter we have evaluated the application of the SDGT to image coding without considering intra-prediction. Since it has been shown that prediction residuals and original signals are very different from a statistical point of view, we leave as future work the study of the application of SDGT for intra-prediction residual coding. Moreover, it would be interesting to study the application of SDGT also for inter-prediction residual coding, in this case SDGT could be used jointly with a segmentation-based motion prediction, such as [80, 81].



(a) Parrots



(b) Lighthouse



(c) Girl

Fig. 5.4 The performance results of the proposed SDGT and DCT 8×8 is presented in term of PSNR values over bitrate.

(a) DCT 8×8

(b) SDGT

Fig. 5.5 A detail on the luminance component of one image compressed with both DCT 8×8 and the proposed SDGT at bitrate of 0.75 bpp.

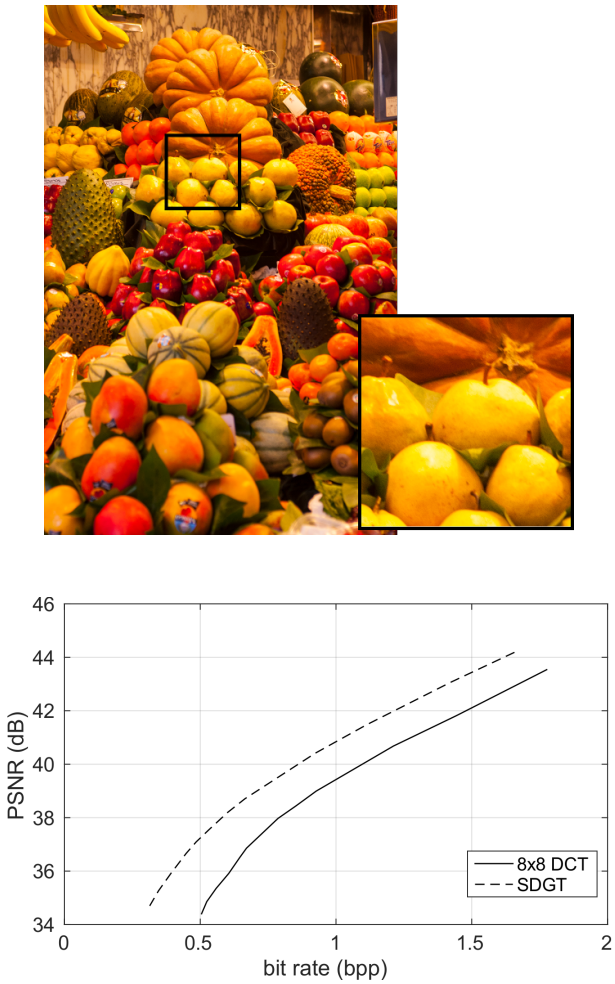


Fig. 5.6 A 2592×3888 sample image with a 512×512 cropped patch (top) and the performance of the proposed SDGT and 8×8 DCT on the cropped region in term of PSNR values over bitrate (bottom).

Chapter 6

Steerable Discrete Cosine Transform

In this chapter, we present a new framework for directional transforms. Starting from the graph transform of a grid graph, we design a new transform, called steerable DCT (SDCT), which can be obtained by steering the 2D-DCT basis in a chosen direction.

6.1 Preliminaries

Before introducing the new SDCT, we first review some elements of graph signal processing, introducing the concept of product graph and discussing the relation between GFT and DCT.

As we have shown in Chapter 2, the 1D-DCT is a valid graph Fourier transform of a path graph $\mathcal{P}_N$. Specifically, the 1D-DCT has N basis vectors $\{\mathbf{v}^{(k)}\}_{k=0}^{N-1}$ which are defined as

$$\mathbf{v}_j^{(k)} = \cos\left(\frac{\pi k}{N}\left(j+\frac{1}{2}\right)\right), \qquad j,k = 0,1,...,N-1. \tag{6.1}$$

Part of the work described in this chapter has been previously published in G. Fracastoro, S. M. Fosson, E. Magli, Steerable Discrete Cosine Transform, *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 303-314, 2017.

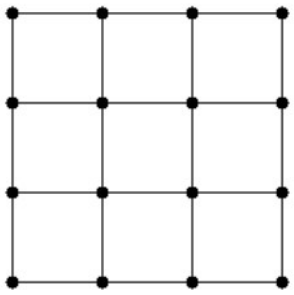


Fig. 6.1 The square grid graph $\mathcal{P}_4 \times \mathcal{P}_4$.

Each $\mathbf{v}^{(k)}$ is the eigenvector of $L(\mathcal{P}_N)$, for any $k = 0, 1, ..., n-1$, L, associated with the eigenvalue

$$\lambda_k = 4\sin^2\left(\frac{\pi k}{2N}\right). \tag{6.2}$$

Given that the multiplicity of the eigenvalues in (6.2) is always equal to 1, the 1D-DCT basis is the unique eigenbasis for $L(\mathcal{P}_N)$, therefore the graph Fourier transform for a signal represented by a path graph is equivalent to the 1D-DCT transform.

Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, let $\mathcal{G} = \mathcal{G}_1 \times \mathcal{G}_2$ be the product graph of $\mathcal{G}_1$ and $\mathcal{G}_2$. Suppose v_1, $v_2 \in \mathcal{V}_1$ and u_1, $u_2 \in \mathcal{V}_2$. Then (v_1, u_1) and (v_2, u_2) are adjacent in $\mathcal{G}$ if and only if one of the following conditions are satisfied [82]: a) $v_1 = v_2$ and $\{u_1, u_2\} \in \mathcal{E}_2$; b) $\{v_1, v_2\} \in \mathcal{E}_1$ and $u_1 = u_2$. Let us now consider the product graph of two path graphs, as shown in Fig. 6.1. If the two path graphs have the same number of vertices, their product graph $\mathcal{P}_n \times \mathcal{P}_n$ is a square grid graph with $N = n^2$ vertices. It has been proved that the basis vectors of the 2D-DCT form an eigenbasis of $L(\mathcal{P}_n \times \mathcal{P}_n)$ [50].

Moreover, the spectrum of the Laplacian of a product graph depends on the spectrum of the two generator graphs, as illustrated in the following theorem.

Theorem 6.1 (Theorem 2.21 in [82]; [83])**.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be graphs on N_1 and N_2 vertices, respectively. Then the eigenvalues of $L(\mathcal{G}_1 \times \mathcal{G}_2)$ are all possible sums of $\lambda_i(\mathcal{G}_1) + \lambda_j(\mathcal{G}_2)$, with $0 \leq i \leq N_1 - 1$ and $0 \leq j \leq N_2 - 1$. Moreover, if $\mathbf{v}^{(i)}$ is an eigenvector of $\mathcal{G}_1$ corresponding to $\lambda_i(\mathcal{G}_1)$, $\mathbf{v}^{(j)}$ an eigenvector of $\mathcal{G}_2$ corresponding to $\lambda_j(\mathcal{G}_2)$, then $\mathbf{v}^{(i)} \otimes \mathbf{v}^{(j)}$ (where $\otimes$ indicates the Kronecker product) is an eigenvector of $\mathcal{G}$ corresponding to $\lambda_i(\mathcal{G}_1) + \lambda_j(\mathcal{G}_2)$.*

6.2 Analysis of the eigenvalues' multiplicity

Leveraging the results presented in the previous section, we build a new transform that can be oriented in any direction. Using Theorem 6.1 and equations (6.1) and (6.2), we can compute the eigenvalues and the eigenvectors of $L(\mathcal{P}_n \times \mathcal{P}_n)$ (which, for simplicity, are labeled with a double index):

$$\lambda_{k,l} = \lambda_k + \lambda_l = 4\sin^2\left(\frac{\pi k}{2n}\right) + 4\sin^2\left(\frac{\pi l}{2n}\right), \tag{6.3}$$

$$\mathbf{v}^{(k,l)} = \mathbf{v}^{(k)} \otimes \mathbf{v}^{(l)}, \quad 0 \leq k, l \leq n-1,$$

where $\mathbf{v}^{(k)}$ is the eigenvector of $\mathcal{P}_n$ corresponding to λ_k and $\mathbf{v}^{(l)}$ is the eigenvector corresponding to λ_l. From (6.3), it is evident that some repeated eigenvalues are present, due to symmetry: $\lambda_{k,l} = \lambda_{l,k}$ for $k \neq l$. Moreover, through straightforward computations, it is possible to prove that the eigenvalue $\lambda = 4$ has algebraic multiplicity $n-1$ and corresponds to all eigenvalues $\lambda_{k,n-k}$ with $1 \leq k \leq n-1$. Therefore, in the spectrum of L there are only $n-1$ eigenvalues with algebraic multiplicity equal to 1 (i.e. $\lambda_{k,k}$ with $k \neq n/2$), and all the others but $\lambda_{k,n-k}$ have algebraic multiplicity 2. It is important to highlight that even if $\lambda_{k,l} = \lambda_{l,k}$ when $k \neq l$, we still have that $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$ are linearly independent, because the Kronecker product is not commutative. Therefore, the geometric multiplicity is equal to the algebraic multiplicity. This means that the dimension of the eigenspaces corresponding to these eigenvalues is bigger than one. This proves the following proposition.

Proposition 6.1. *The 2D-DCT is not the unique eigenbasis of the Laplacian matrix of a square grid graph.*

In Fig. 6.2 the 2D-DCT basis with $n = 8$ is represented in matrix form; as an example, we have highlighted in red the corresponding two eigenvectors of an eigenvalue with multiplicity 2: we can see that they are clearly related to each other, since they represent the same frequency, one in the horizontal direction and the other in the vertical direction.

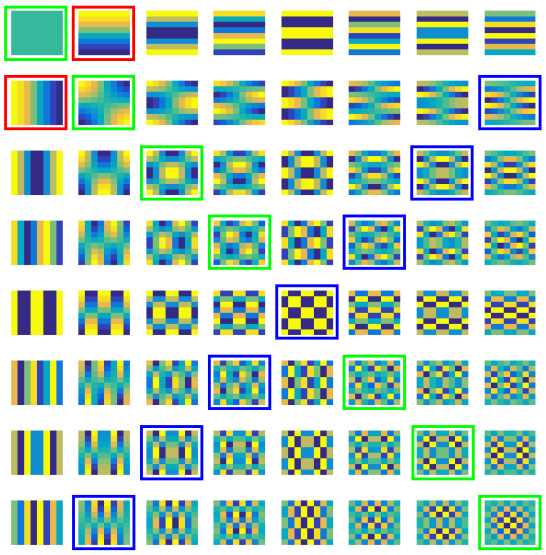


Fig. 6.2 2D-DCT basis vectors represented in matrix form (with $n = 8$): the corresponding two eigenvectors of an eigenvalue with multiplicity 2 are highlighted in red, the $n-1$ eigenvectors corresponding to $\lambda = 4$ are highlighted in blue and the $n-1$ eigenvectors corresponding to the eigenvalues with algebraic multiplicity 1 are highlighted in green.

6.3 Transform definition

Since the 2D-DCT is not the unique eigenbasis for $L(\mathcal{P}_n \times \mathcal{P}_n)$, we aim to find all the other possible eigenbases and choose as transform matrix the one that better fits the properties of the signal that we want to process.

Given an eigenvalue $\lambda_{k,l}$ of $L(\mathcal{P}_n \times \mathcal{P}_n)$ with multiplicity 2 and the two vectors of the 2D-DCT $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$ that are the eigenvectors of $L(\mathcal{P}_n \times \mathcal{P}_n)$ corresponding to $\lambda_{k,l}$, we can write any other possible basis of the eigenspace corresponding to $\lambda_{k,l}$ as the result of a rotation of $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$

$$\begin{bmatrix} \mathbf{v}^{(k,l)'} \\ \mathbf{v}^{(l,k)'} \end{bmatrix} = \begin{bmatrix} \cos\theta_{k,l} & \sin\theta_{k,l} \\ -\sin\theta_{k,l} & \cos\theta_{k,l} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(k,l)} \\ \mathbf{v}^{(l,k)} \end{bmatrix}, \tag{6.4}$$

where $\theta_{k,l}$ is an angle in $[0, 2\pi]$. The rotation described in (8.4) can also be defined as a Givens rotation [84] in the plane described by $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$ of the n^2-dimensional space.

For every $\lambda_{k,l}$ with multiplicity 2, we can rotate the corresponding eigenvectors as shown in (8.4); the $n-1$ eigenvectors corresponding to $\lambda = 4$ are rotated in pairs $\mathbf{v}^{(k,n-k)}$ and $\mathbf{v}^{(n-k,k)}$, if n is even $\mathbf{v}^{(\frac{n}{2},\frac{n}{2})}$ is not rotated. In the 2D-DCT matrix, the pairs $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$ are replaced with the rotated ones $\mathbf{v}^{(k,l)'}$ and $\mathbf{v}^{(l,k)'}$ obtaining a new transform matrix $V(\theta) \in \mathbb{R}^{n^2 \times n^2}$ that can be defined only by the rotation angles used, which we have to transmit to the decoder. The number of angles used is equal to the number of rotated pairs, that is $p = \frac{n(n-1)}{2}$. The new transform matrix $V(\theta)$ can be written as

$$V(\theta) = V\mathrm{R}(\theta),$$

where $V = V(0) \in \mathbb{R}^{n^2 \times n^2}$ is the 2D-DCT transform matrix, $\theta \in \mathbb{R}^p$ is the vector containing all the angles used and $\mathrm{R}(\theta) \in \mathbb{R}^{n^2 \times n^2}$ is the rotation matrix, whose structure is defined so that, for each pair of vectors, it performs the rotation as defined in (8.4).

$\mathrm{R}(\theta)$ can be decomposed in two matrices as

$$\mathrm{R}(\theta) = \Delta + \widetilde{\mathrm{R}}(\theta),$$

where $\Delta \in \mathbb{R}^{n^2 \times n^2}$ is a constant matrix representing the vectors that do not rotate, and $\widetilde{\mathrm{R}}(\theta) \in \mathbb{R}^{n^2 \times n^2}$ represents the vectors that are rotated. Δ is a diagonal matrix, with $\Delta_{ii} = 1$ for any $i = kn + k$ with $0 \leq k \leq n-1$; otherwise, $\Delta_{ii} = 0$. Given $0 \leq k, l \leq n-1$ and $k \neq l$, if $i = kn + l$ and $j = ln + k$, then $\widetilde{\mathrm{R}}(\theta)_{ii} = \widetilde{\mathrm{R}}(\theta)_{jj} = \cos\theta_{k,l}$, $\widetilde{\mathrm{R}}(\theta)_{ij} = \sin\theta_{k,l}$ and $\widetilde{\mathrm{R}}(\theta)_{ji} = -\sin\theta_{k,l}$, otherwise $\widetilde{\mathrm{R}}(\theta)_{ij} = 0$. Then, for any signal $\mathbf{x} \in \mathbb{R}^{n^2}$ our new transform, which in the following will be referred to as SDCT, is defined as follows:

$$\hat{\mathbf{x}} = V(\theta)^T \mathbf{f} = \mathrm{R}(\theta)^T V^T \mathbf{f} = (\Delta^T + \widetilde{\mathrm{R}}(\theta)^T) V^T \mathbf{x}. \tag{6.5}$$

Equation (6.5) shows that the SDCT can be decomposed as a product of a rotation matrix $\mathrm{R}(\theta)$ and the 2D-DCT transform matrix V. Moreover, let $\hat{\mathbf{x}}_{DCT} \in \mathbb{R}^{n^2}$ be the DCT coefficients of the signal $\mathbf{x}$, then the SDCT can be computed in the following way

$$\hat{\mathbf{x}} = \mathrm{R}(\theta)^T \hat{\mathbf{x}}_{DCT}. \tag{6.6}$$

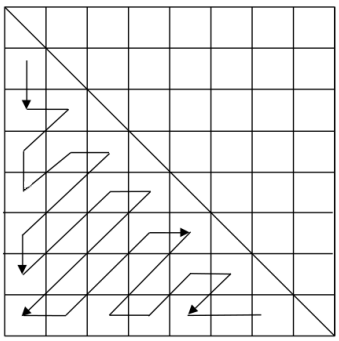


Fig. 6.3 Zigzag ordering for the p components of θ.

In this way, the complexity of the SDCT can be drastically reduced because $\hat{\mathbf{x}}_{DCT}$ can be computed using the separability property. Then, to compute the SDCT coefficients, $\hat{\mathbf{x}}_{DCT}$ is multiplied by the sparse matrix $\mathrm{R}(\theta)$.

The components $\theta_{k,l}$ of θ are ordered using the zigzag pattern shown in Fig. 6.3. Unlike the classical zigzag ordering, in this case we consider only p elements, since $\theta_{k,l} = \theta_{l,k}$ and the diagonal elements $\theta_{k,k}$ are not considered, since the eigenvectors $\mathbf{v}^{(k,k)}$ do not rotate.

The transform (6.5) is still the graph transform of a square grid graph, but with a different set of orientations with respect to DCT. As an example, in Fig. 6.4, we show the basis vectors obtained rotating by $\frac{\pi}{4}$ every pair of eigenvectors. As can be seen, the diagonal elements $\mathbf{v}^{(k,k)}$ are the same as the DCT ones because the corresponding eigenvalues have multiplicity one, instead all the others are rotated by $\frac{\pi}{4}$.

6.4 Probabilistic interpretation of the SDCT

It is well known that the 1D-DCT is close to the optimum KLT for Markov sources with high correlation coefficient [1]. Instead, the optimality of the 2D-DCT has been less studied than the 1D case. In [50], the authors have shown that if we consider a first order Gaussian Markov random field with correlation coefficient equal to one (the corresponding graph is a uniform grid such as the one shown in Fig. 6.1), then the 2D-DCT achieves optimal signal decorrelation. They have also pointed out that it is not the unique transform that achieves this result, because the eigenvector matrix is not unique. The

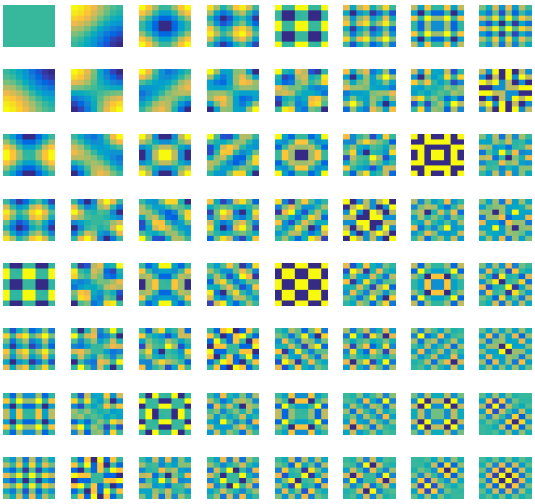


Fig. 6.4 Steerable DCT with $\theta = \frac{\pi}{4}$.

SDCT represents all the possible optimal transforms for this type of signals. In the next chapter, we use a determistic approach in order to define for each block the best orientation for the SDCT. Instead, we leave as future work a probabilistic interpretation of the SDCT, where we could investigate what is the influence of the transform orientation on its optimality.

Chapter 7

SDCT: application to image and video compression

In the previous chapter, we have shown that a new directional transform, called SDCT, can be derived rotating $2p = n(n-1)$ columns of an $n^2 \times n^2$ DCT matrix. In this chapter we present a few image compression algorithms based on the SDCT. These algorithms explore different possibilities for the choice of the rotation angles. First, we consider the case where we have only one rotation angle per block, rotating all the eigenspaces by the same angle. Then, we exploit the possibility of subdiving the eigenspaces in subbands and use one rotation angle per subband. Finally, we also present the case where for each eigenspace we use its corresponding optimal angle.

7.1 SDCT-1

In principle, we can use more than one rotation angle per block. However, in this case the cost required to transmit all the rotation angles may be too high. For this reason, we first consider the case where we use the same rotation angle

Part of the work described in this chapter has been previously published in G. Fracastoro, E. Magli, Steerable Discrete Cosine Transform, *Proc. of IEEE International Workshop on Multimedia Signal Processing*, 2015, pp 1-6; G. Fracastoro, E. Magli, Subspace-sparsifying Steerable Discrete Cosine Transform from Graph Fourier Transform, *Proc. of IEEE International Conference on Image Processing*, 2016, pp. 1534-1538; G. Fracastoro, S. M. Fosson, E. Magli, Steerable Discrete Cosine Transform, *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 303-314, 2017.

Algorithm 2 SDCT-1. θ: set of p angles between 0° and 90°, J: objective function, J_{opt}: optimal value of the objective function, V: 2D DCT matrix.

```
Set J_opt to 0;
Set the optimal angle to 0;
for i=1 to p do
  Rotate each pair of vectors in V by θ_i;
  Build a new transform matrix V' with the rotated vectors;
  Compute J(θ_i): the value of the objective function J using V';
  if J(θ_i) > J_opt then
    Set the optimal angle to θ_i;
    J_opt = J(θ_i);
  end if
end for
```

for every eigenspace, transmitting, therefore, only one rotation angle per block. Rotating all the eigenspaces by the same angle, we obtain a new transform matrix, that we call SDCT-1, which is still the graph transform of a square grid graph, but its orientation is different from that of the DCT.

The aim of using a transform matrix whose vector basis has a different orientation from the horizontal/vertical one is to obtain a more compact signal representation by unbalancing the energy of the transform coefficients. For each pair of rotated eigenvectors, the total energy of the corresponding transform coefficients remains unchanged, but it is possible to sparsify the signal representation in each eigenspace. In the optimal case, the rotation compacts all the energy of the pair in one of the two coefficients. If every pair of vectors is rotated by the same angle, in the majority of cases we will not achieve a complete unbalancing that zeros out one of the two coefficients, but we will still obtain an improvement in the compression performance.

To choose the best rotation angle, we use an exhaustive method, finding, among a finite set of p possible angles between 0° and 90°, the one that optimizes a predetermined objective function J (e.g a measure of the sparsity of the transform coefficients, more details will be given in the following section). Algorithm 2 describes the method used.

(a) House (b) Boat (c) Lena

Fig. 7.1 Original images.

7.1.1 Experimental Results

For the purpose of experimentation, first we subdivide the image into blocks; then, in each block we apply the SDCT-1. To evaluate the performance of the proposed transform, given an image $\mathbf{x} \in \mathbb{R}^N$ we use the M term non-linear approximation, where we keep the M largest coefficients (where $M < N$) and set the others to zero:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{M} \hat{x}_i \mathbf{v}_i,$$

where $\tilde{\mathbf{x}}$ is the reconstructed image, $\{\hat{x}_i\}_{1 \leq i \leq M}$ are the M transform coefficients with largest magnitude and $\mathbf{v}_i$ are the corresponding transform basis vectors. To find the best rotation angle, we choose, for each M, the one that maximizes the energy in the M largest coefficients $\left(J = \sum_{i=1}^{M} \hat{x}_i^2\right)$. Then, we compute the PSNR of each image and we compare the performance of the proposed SDCT-1 with the standard DCT. We have tested this method on several grayscale images, three of them are shown in Fig. 7.1.

The results presented are preliminary and they are meant to demonstrate the potentiality of the proposed transform, but further work is needed to develop image coding applications. In particular, the non-linear approximation used does not take into account the overhead bits needed to transmit the rotation angle, therefore the comparison with the DCT is not completely fair, even if the required overhead for the SDCT-1 is very low.

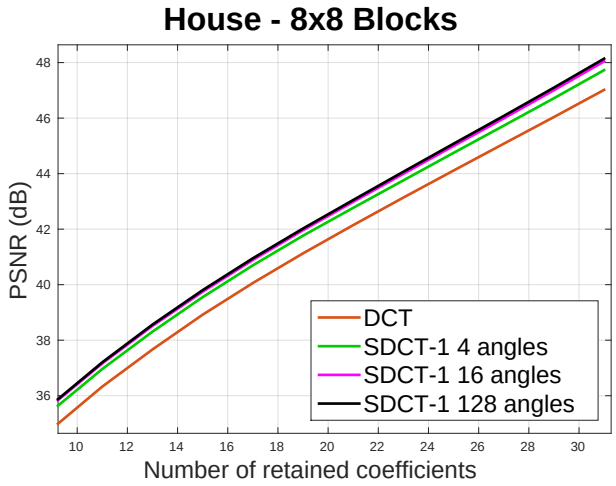


Fig. 7.2 M-term non-linear approximation using different angle quantizations.

Angle quantization

To obtain a finite set of angles, we perform an uniform quantization of the angles between 0° and 90°. Since the SDCT-1 needs to transmit as side information the rotation angle that we used for the transform, it is important to have a small number of possible angles. We have evaluated the performance of the SDCT-1 as a function of the number of available angles. The results obtained using 8×8 blocks are shown in Fig. 7.2. From the results, we can see that increasing the number of angles improves the performance of the SDCT-1, but after a while this improvement becomes not significant, for example the performance using 16 angles and 128 angles is nearly the same. For the tests presented in the following sections, we decided to use quantization onto 16 possible angles.

Effect of block size

The proposed SDCT-1 can be applied to square blocks of any size. However, in our experiments we have noticed that the gain of the SDCT-1 depends on the block size used. In Fig. 7.3 we show the performance curves obtained with different block sizes. On average, we have a coding gain of 1.5 dB with 4×4 blocks, 0.7 dB with 8×8 blocks and 0.25 dB with 16×16 blocks. We can clearly see that the effectiveness of the SDCT-1 increase when we use small

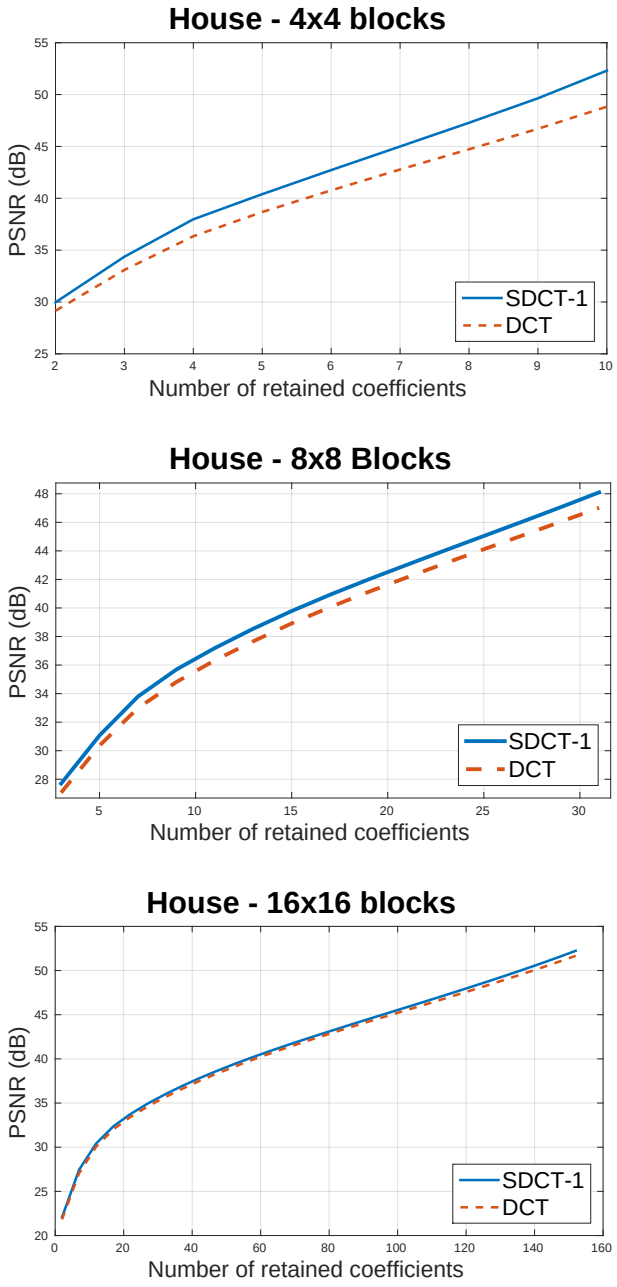
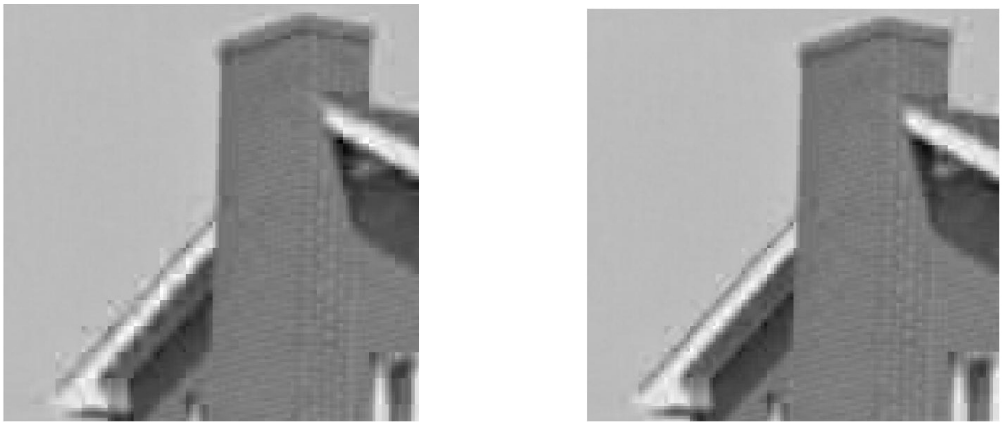


Fig. 7.3 M-term non-linear approximation using different block sizes.

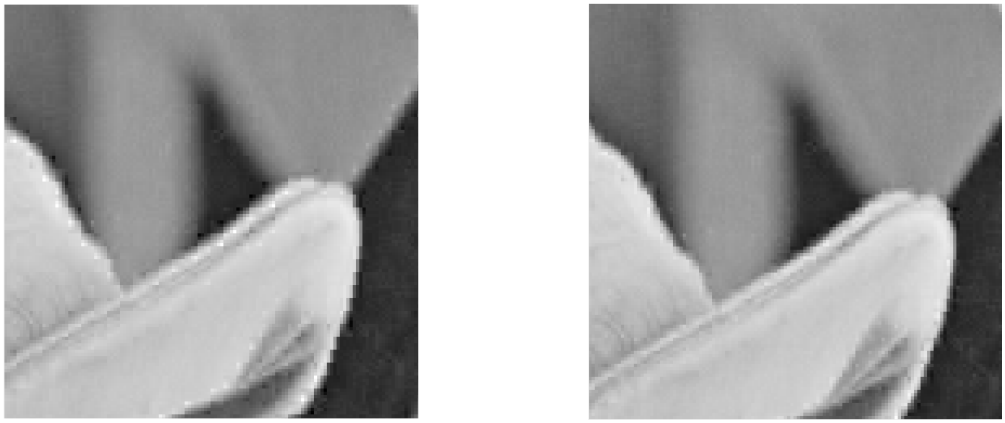
blocks, because in larger blocks it is more difficult to find just one predominant direction.

A detail of the significant improvement obtained by the SDCT-1 can be visually appreciated in Fig. 7.4 and 7.5. As can be seen, the SDCT-1 provides



(a) DCT (b) SDCT-1

Fig. 7.4 Visual comparison between the conventional DCT and the proposed SDCT-1: a detail of the House image reconstructed from $M = 6$ coefficients using 8×8 blocks.



(a) DCT (b) SDCT-1

Fig. 7.5 Visual comparison between the conventional DCT and the proposed SDCT-1: a detail of the Lena image reconstructed from $M = 3$ coefficients using 4×4 blocks.

much better visual quality, minimizing artifacts along the edges thanks to the alignment of the transform to the edge direction.

Comparison with DDCT

In the case of 8×8 blocks, we have also made a comparison with the DDCT presented in [21]. In Fig. 7.6, the results obtained with some test images are shown. We can see that the performances of the two methods are very similar,

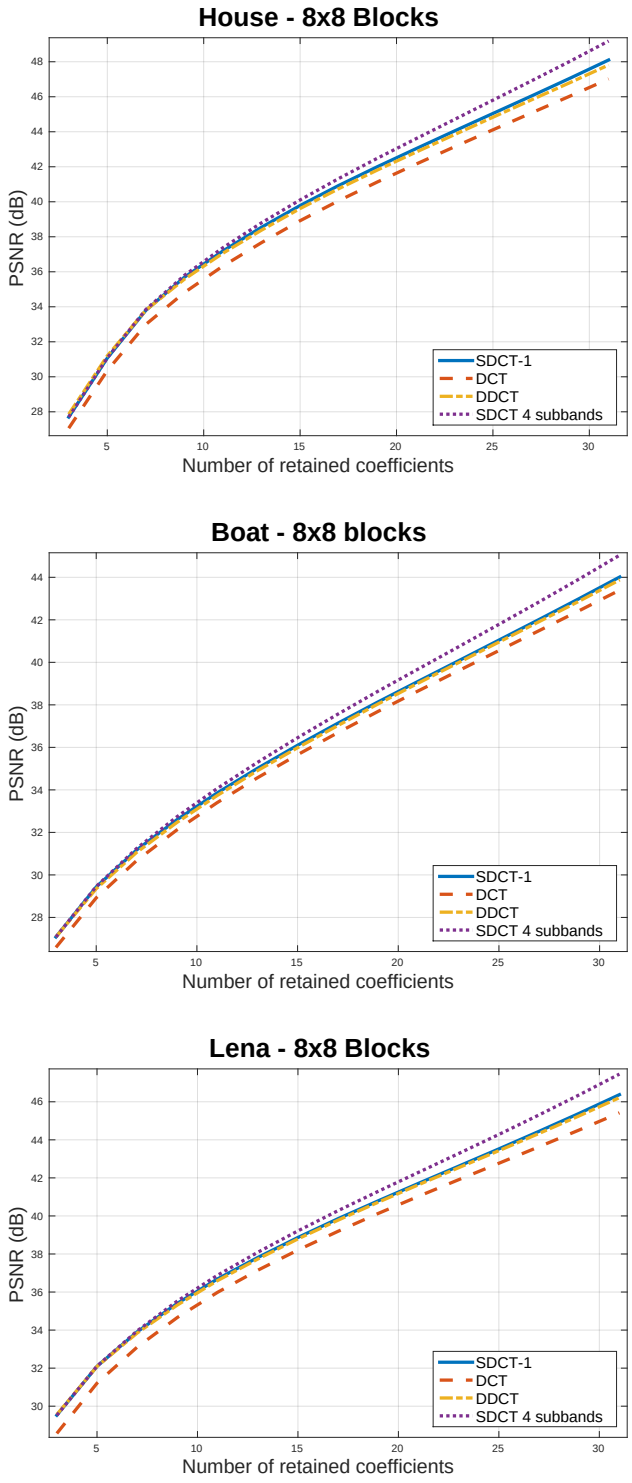


Fig. 7.6 Performance comparison between DCT, SDCT-1 and DDCT.

even if the proposed SDCT-1 is slightly better with an average coding gain of approximately 0.1 dB. Even if the performances of the two transforms are similar, the SDCT-1 presents a more general framework for image coding that can be further extended using more than one rotation angle. In order to show the potential of the proposed framework, we have ordered the pairs of vectors using the zigzag ordering and divided them in four subbands of equal size, then we have used a different rotation angle for each subband. The results obtained are plotted in Fig. 7.6. We can see that the improvement over the SDCT-1 and the DDCT is significant, with an average quality gain of 0.45 dB over the SDCT-1 and of 1.15 dB over the standard DCT.

7.2 Rate-distortion optimization

In the previous section, we have presented a new algorithm for image compression based on the SDCT, where the 2D-DCT basis is rotated by a single given angle for each image block.

In this section, we analyse the broader of finding the best set of rotations of the 2D-DCT basis for each image block. Considering more than one angle per block, we can potentially obtain a more compact representation at the price of more side information to transmit. The tradeoff can be analysed from a RD perspective. We first cast the problem as the problem as the minimization of a RD functional. The minimum provides the optimal number of rotation angles per block as well as the angles' values. The problem is well-posed (the global minimum exists), but it is non-convex, hence finding the global minimum is non trivial. The best feasible strategy that one can conceive in such case is iterative alternated minimization, that allows to get to a local minimum or a saddle point. This is the basis of our first proposed algorithm, named steerable DCT through alternated minimization (SDCT-AM). If suitably initialized, SDCT-AM is proved to always outperform DCT in RD terms. We have also investigated other strategies to define and transmit the angles' distribution, in order to reduce the angles' transmission cost, and propose a subdivision into subbands that can be encoded as a binary tree. This is the key idea for our second proposed algorithm, named SDCT-BT, which significantly decreases the amount of side information.

7.2.1 RD model

Let $V(\theta) = V\mathrm{R}(\theta)$ be the steered transform matrix, $\mathbf{x} \in \mathbb{R}^{n^2}$ be the image block, $\theta = (\theta_1, \ldots, \theta_p)$ be the ordered set of angles, and $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_{n^2})^T$ be the coefficients of the transform (6.5). As a distortion metric we employ the reconstruction error:

$$\mathcal{D}(\hat{\mathbf{x}}, \theta) := \|\mathbf{x} - V(\theta)\hat{\mathbf{x}}\|_2^2. \tag{7.1}$$

We consider two rate contributions, that is, the transform coefficients rate $\mathcal{R}_{\hat{\mathbf{x}}}$ and the rotation angles' rate $\mathcal{R}_\theta$. The total rate is $\mathcal{R}(\hat{\mathbf{x}}, \theta) = \mathcal{R}_{\hat{\mathbf{x}}} + \mathcal{R}_\theta$.

In [85, 25, 62], it has been shown that for DCT transforms there is an approximately linear relationship between the coding bitrate $\mathcal{R}_{\hat{\mathbf{x}}}$ and the ℓ_0-norm of $\hat{\mathbf{x}}$, that is, the number of its non-zero coefficients, i.e.

$$\mathcal{R}_{\hat{\mathbf{x}}} = \alpha \|\hat{\mathbf{x}}\|_0 \tag{7.2}$$

where α can be empirically found [85].

Let us now discuss $\mathcal{R}_\theta$. In the previous section, we have considered the simple case of using the same angle for all the eigenspaces, and concluded that this is sufficient to outperform classical 2D-DCT. Our aim is now to study the intermediate cases, seeking the optimal number and values of angles yielding the best balance between recovery accuracy and rate.

Specifically, we split the angles into subbands of DCT coefficients, choosing a single angle for all coefficients in each subband, so that the vector θ is piecewise constant. Let s be the number of subbands: if $s < p$, instead of transmitting p angles, we require only s angle values and s indexes indicating where the subvectors end. Assuming no compression for the angles and a quantization over q_θ values in $[0, 2\pi]$ for each angle, the transmission amounts to $s\lceil \log_2 q_\theta \rceil + s\lceil \log_2 p \rceil$, which clearly increases much slower than than $p\lceil \log_2 q_\theta \rceil$. We notice that s can be expressed as a function of θ as follows:

$$s = \|B\theta\|_0$$

where $B \in \mathbb{R}^{p \times p}$ is the discrete difference operator, given by:

$$B = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & -1 & 1 \end{pmatrix}.$$

In conclusion, we define the angles rate as follows:

$$\mathcal{R}_\theta = \|B\theta\|_0(\lceil \log_2 q_\theta \rceil + \lceil \log_2 p \rceil). \tag{7.3}$$

Finally, we assume that both $\hat{\mathbf{x}}$ and θ are quantized, and denote as $\mathrm{Q}_{\hat{\mathbf{x}}} \subset \mathbb{R}$ and $\mathrm{Q}_\theta \subset [0, \pi]$ the respective sets of available reconstruction values for each component, so that $\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^{n^2}$ and $\theta \in \mathrm{Q}_\theta^p$. We are now ready to define our RD optimization problem. As in [86], we consider the following Lagrangian relaxation:

$$\begin{aligned} &\min_{\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^N,\ \theta \in \mathrm{Q}_\theta^p} J(\hat{\mathbf{x}}, \theta) \\ &J(\hat{\mathbf{x}}, \theta) = \mathcal{D}(\hat{\mathbf{x}}, \theta) + \gamma(\mathcal{R}_{\hat{\mathbf{x}}} + \mathcal{R}_\theta) \\ &\qquad = \|\mathbf{x} - V(\theta)\hat{\mathbf{x}}\|_2^2 + \\ &\qquad + \gamma\left[\alpha\|\hat{\mathbf{x}}\|_0 + (\lceil \log_2 q_\theta \rceil + \lceil \log_2 p \rceil)\|B\theta\|_0\right], \end{aligned} \tag{7.4}$$

where $\gamma > 0$ is the Lagrangian parameter.

The problem (7.4) is similar to sparse signal recovery problems, for which hard thresholding techniques can be used [87]. Briefly, a functional of kind $\|A\mathbf{w} - \mathbf{z}\|_2^2 + \gamma\|\mathbf{w}\|_0$ with $\mathbf{w} \in \mathbb{R}^n$ and invertible $A \in \mathbb{R}^{n \times n}$ has global minimum at $\mathcal{H}_{\sqrt{\gamma}}[A^{-1}\mathbf{z}]$, where $\mathcal{H}_{\sqrt{\gamma}} : \mathbb{R}^n \to \mathbb{R}^n$ is the hard-thresholding operator that sets to zero all the components smaller than $\sqrt{\gamma}$ in magnitude of its input vector. This can be derived as a simpler subcase of iterative hard thresholding for sparse problems [87, Equation 2.1-2.2]: since our transform matrix is orthogonal, the procedure stops after one iteration.

Our problem is made more difficult by the non-convexity of the distortion term due to the variable θ. However, we remark that the problem is well posed, because it is lower bounded by 0, and it is proper (if $\hat{\mathbf{x}}$ goes to infinity, J

tends to infinity as well). This encourages to search a solution; to this end, we undertake alternated minimization on separated variables. In particular, we notice that the problem can be analytically solved with respect to the individual variables $\hat{\mathbf{x}}$ and $\theta_1, \ldots, \theta_p$.

7.2.2 Proposed algorithms for RD optimization

In this section, we present the proposed algorithms SDCT-AM and SDCT-BT to seek the best set of rotations for SDCT.

In the previous section, we have defined the RD optimization problem (7.4) and observed that a global solution is difficult to find due to global non-convexity. However, the problem is mathematically tractable in the individual variables $\hat{\mathbf{x}}, \theta_1, \theta_2, \ldots, \theta_q$, as we are going to show, and an alternated minimization achieves a partial optimum (i.e., a local minimum or a saddle point). This is the basis of SDCT-AM.

Alternated minimization: SDCT-AM

Assuming θ fixed, the evaluation of $\min_{\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^N} J(\hat{\mathbf{x}}, \theta)$ is straightforward. We have

$$\begin{aligned} \min_{\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^N} J(\hat{\mathbf{x}}, \theta) &= \min_{\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^N} \mathcal{D}(\hat{\mathbf{x}}, \theta) + \gamma\alpha \|\hat{\mathbf{x}}\|_0 \\ &= \min_{\hat{\mathbf{x}} \in \mathrm{Q}_{\hat{\mathbf{x}}}^N} \sum_{i=1}^{n^2} \left[\hat{x}_i - (V^T(\theta)\mathcal{I})_i\right]^2 + \gamma\alpha \left\|\hat{x}_i\right\|_0 . \end{aligned}$$

Therefore, we can solve a separated problem for each component $\hat{x}_i$, whose solution is given by

$$\mathcal{H}_{\sqrt{\gamma\alpha}}\left[\mathcal{Q}\left[(V^T(\theta)\mathbf{x})_i\right]\right]$$

where, for any $z \in \mathbb{R}$, $\mathcal{Q}[z]$ and $\mathcal{H}_{\sqrt{\gamma\alpha}}[z]$ respectively indicate the quantization operator that projects onto $\mathrm{Q}_{\hat{\mathbf{x}}}$ and the hard thresholding operator with threshold $\sqrt{\gamma\alpha}$ defined as $\mathcal{H}_{\sqrt{\gamma\alpha}}[z] = x$ if $|z| > \sqrt{\gamma\alpha}$, and $\mathcal{H}_{\sqrt{\gamma\alpha}}[z] = 0$ if $|z| \leq \sqrt{\gamma\alpha}$.

We notice that

$$\left[\mathcal{Q}\left[(V^T(\theta)\mathbf{x})_i\right]\right] = \arg\min_{\hat{x}_i \in \mathrm{Q}_{\hat{\mathbf{x}}}} \left[\hat{x}_i - (V^T(\theta)\mathbf{x})_i\right]^2$$

since $\left[\hat{x}_i - (V^T(\theta)\mathbf{x})_i\right]^2$ is convex and symmetric.

The procedure to minimize $J(\hat{\mathbf{x}},\theta)$ with respect to θ_j, $j \in \{1,\ldots,p\}$, is similar. We have

$$\min_{\theta_j \in \mathrm{Q}_\theta} J(\hat{\mathbf{x}},\theta) = \min_{\theta_j \in \mathrm{Q}_\theta} \mathcal{D}(\hat{\mathbf{x}},\theta) + \gamma(\lceil \log_2 q_\theta \rceil + \lceil \log_2 p \rceil) \left\| B\theta \right\|_0$$

where the term $\left\| B\theta \right\|_0$ can be substituted by $\left\| \theta_j - \theta_{j+1} \right\|_0 + \left\| \theta_j - \theta_{j-1} \right\|_0$ for $j \in \{2,\ldots,p-1\}$, by $\left\| \theta_1 \right\|_0 + \left\| \theta_1 - \theta_2 \right\|_0$ for $j = 1$, and by $\left\| \theta_q - \theta_{q-1} \right\|_0$ for $j = p$.

First, we analytically evaluate $\min_{\theta_j \in [0,2\pi]} \mathcal{D}(\hat{\mathbf{x}},\theta)$. Since $V(\theta)$ is orthogonal for any θ,

$$\|\mathbf{x} - V(\theta)\hat{\mathbf{x}}\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{x}^T V(\theta)\hat{\mathbf{x}} + \|\hat{\mathbf{x}}\|_2^2.$$

Furthermore, it is straightforward to check that we can define a matrix $U = U(\hat{\mathbf{x}}) \in \mathbb{R}^{n^2 \times 2p}$ such that

$$V\widetilde{\mathrm{R}}(\theta)\hat{\mathbf{x}} = U(\hat{\mathbf{x}})(\cos(\theta_1), \sin(\theta_1), \ldots, \cos(\theta_p), \sin(\theta_p))^T,$$

$\widetilde{\mathrm{R}}(\theta)$ being defined in Section 6.3. In this way,

$$\begin{aligned} V(\theta)\hat{\mathbf{x}} &= V[\Delta + \widetilde{\mathrm{R}}(\theta)]\hat{\mathbf{x}} \\ &= V\Delta\hat{\mathbf{x}} + U(\hat{\mathbf{x}})(\cos(\theta_1), \sin(\theta_1), \ldots)^T. \end{aligned}$$

Therefore,

$$\min_{\theta_j \in [0,\pi]} \mathcal{D}(\hat{\mathbf{x}},\theta) = \min_{\theta_j \in [0,\pi]} -2\mathbf{x}^T U(\hat{\mathbf{x}})(\cos(\theta_1), \sin(\theta_1), \ldots)^T.$$

We then compute the derivative with respect to θ_j, which is equal to zero when $\mathbf{x}^T(U^{(2j)}\sin(\theta_j) - U^{(2j+1)}\cos(\theta_j)) = 0$, i.e.,

Algorithm 3 SDCT-AM

Initialize: $\theta(0), \hat{\mathbf{x}}(0)$;
for t=1,2,... **do**
  $\hat{\mathbf{x}}(t) = \arg\min_{\hat{\mathbf{x}} \in \mathrm{Q}(\hat{\mathbf{x}})^N} J(\hat{\mathbf{x}}, \theta)$ (see Section 7.2.2)
  for $j = p, p-1, \ldots, 1$ **do**
    $\theta_j(t) = \arg\min_{\theta_j \in \mathrm{Q}(\theta)} J(\hat{\mathbf{x}}, \theta)$ (see Section 7.2.2)
  end for
  if $J(\hat{\mathbf{x}}(t-1), \theta(t-1)) = J(\hat{\mathbf{x}}(t), \theta(t))$ **then**
    break
  end if
end for

$$\theta_j = \arctan\left(\frac{\mathbf{x}^T U^{(2j+1)}}{\mathbf{x}^T U^{(2j)}}\right)$$

where $U^{(i)}$ indicates the ith column of U. This equation has one solution in $[0, \pi]$, which could be either the maximum or the minimum. For continuity, it suffices to compare this solution with the extreme values $\theta_j = 0$ and $\theta_j = \pi$ to obtain the minimum.

Afterwards, as for $\hat{x}_i$, we proceed by projecting onto Q_{θ_j} (again, convexity and symmetry of the subproblem guarantee that $\widehat{\theta}_j = \arg\min_{\theta_j \in \mathrm{Q}_{\theta_j}} \mathcal{D}(\hat{\mathbf{x}}, \theta) = \mathcal{Q}[\arg\min_{\theta_j \in [0,\pi]} \mathcal{D}(\hat{\mathbf{x}}, \theta)]$). Finally, we perform hard thresholding, which consists in evaluating which one among $\widehat{\theta}_j, \theta_{j-1}, \theta_{j+1}$ is the most convenient choice for θ_j, $j = 2, \ldots, p-1$ that is, which value provides the minimum J. For $j = 1$ and $j = p$, clearly the choice is among $\widehat{\theta}_1, 0, \theta_2$, and $\widehat{\theta}_p, \theta_{p-1}$.

Alternating these minimization tasks we obtain SDCT-AM, which is summarized in Algorithm 3.

Theorem 7.1. *There is a time t_0 in which $J(\hat{\mathbf{x}}(t), \theta(t))$ in SDCT-AM stabilizes at a partial optimum.*

Proof. The alternated minimization of SDCT-AM guarantees that the sequence $J(\hat{\mathbf{x}}(t), \theta(t))$ is not increasing. Since J is lower bounded by 0 and is a proper function (if $\hat{\mathbf{x}}$ goes to infinity, J tends to infinity), it admits a minimum. Therefore $J(\hat{\mathbf{x}}(t), \theta(t))$ is not increasing and compact, which implies that is convergent. Since $\hat{\mathbf{x}}(t)$ and $\theta(t)$ are quantized values, convergence turns out to be a stabilization, that is, from a time step t_0, $J(\hat{\mathbf{x}}(t), \theta(t))$ is constant. Finally,

it is easy to check that $(\hat{\mathbf{x}}(t_0), \theta(t_0))$ is partial optimum, because the functional increases moving along the coordinate directions.

□

A consequence of this theorem is that the SDCT-AM performance is always better than or equal to the DCT performance, in RD terms. In fact, Since SDCT-AM decreases J, it is sufficient to initialize SDCT-AM with DCT to be sure to perform better (or at least equivalently, in the case that DCT is a partial optimum of J).

Moreover, the theorem suggests also a stop criterion for SDCT-AM: when $J(\hat{\mathbf{x}}(t), \theta(t)) = J(\hat{\mathbf{x}}(t-1), \theta(t-1))$, the algorithm can be stopped.

Binary tree for angles structure: SDCT-BT

SDCT-AM (Algorithm 3) is proved to achieve a partial optimum of the RD functional J, which is the best results that one can expect to achieve, due to the non-convexity of the problem. In the following we propose an alternative algorithm, called SDCT-BT, which reduces the angles side information cost, allowing more freedom in choosing the rotation angles. Based on the construction of a binary tree to describe the angles subband division, SDCT-BT cannot be theoretically analyzed in terms of a minimization problem, but is experimentally proved to perform well.

Before illustrating SDCT-BT, we specify that in this approach $\hat{\mathbf{x}}$ and θ are no more considered as separated variables, since $\hat{\mathbf{x}}$ in this case is the vector of the quantized transform coefficients obtained by performing the SDCT: each time we modify θ, we automatically set $\hat{\mathbf{x}} = \mathcal{Q}[V(\theta)^T \mathbf{x}]$, where $\mathcal{Q}$ indicate the operation of quantization onto $\mathrm{Q}_{\hat{\mathbf{x}}}$. Therefore, we will only use the variable θ, and accordingly we will use $J(\theta)$ to indicate the cost functional.

Moreover, $J(\theta)$ is slightly different from $J(\hat{\mathbf{x}}, \theta)$ in the rate definition. For $R_{\hat{\mathbf{x}}}$, we use the real bitrate, while R_θ is determined by the angle selection procedure that we illustrate in the following.

The angles setting of SDCT-BT is as follows. We start from a single angle value, say one subband, and we iteratively decide if it is convenient to split into different subbands. Specifically, we impose that each subband can be divided

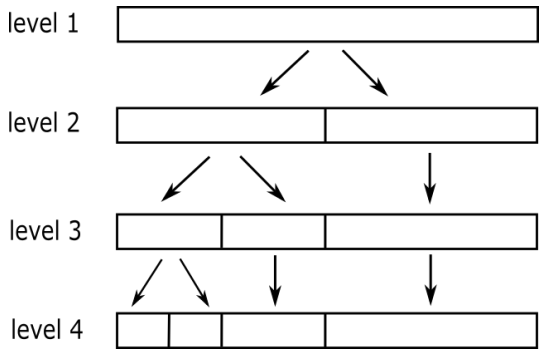


Fig. 7.7 Binary subband subdivision for SDCT: from level 1 downwards, we split a subband if this operation decreases the cost functional J

into two subbands of equal length if this decreases J (spare pairs of vectors are included in the last group), as shown in Fig. 7.7. The decision about splitting a subband is taken by performing an exhaustive search over all possible q_θ angles and selecting the one minimizing J; if the so-obtained $J(\theta)$ is smaller than the current cost $\widehat{J}$, then the split is accepted, and $\widehat{J} = J(\theta)$. We proceed until no more improvement can be obtained, or when the maximum number of subbands is achieved.

As depicted in Fig. 7.8, this procedure is efficient because it can be encoded as a binary decision tree with the root set at level 1. Each node of this tree represents a possible subband and is set to 1 if it actually is a subband, and 0 otherwise. Nodes labeled with 0 are linked to two new nodes, while nodes labeled with 1 are leafs. We represent the final subband subdivision by signaling the decision tree starting from top level 1.

In this way, if the number of subbands is s, the number of nodes in the decision tree is $2s-1$; then we have to signal only $2s-1$ bits. For SDCT-AM the subband structure is encoded over $s\lceil \log_2 p \rceil = \|B\theta\|_0 \lceil \log_2 p \rceil$, which is larger than $2s-1$ for any $p \geq 2$.

SDCT-BT is summarized in Algorithm 4. As one can deduce from Fig. 7.8, for each accepted split we use 2 additional bits to signal it.

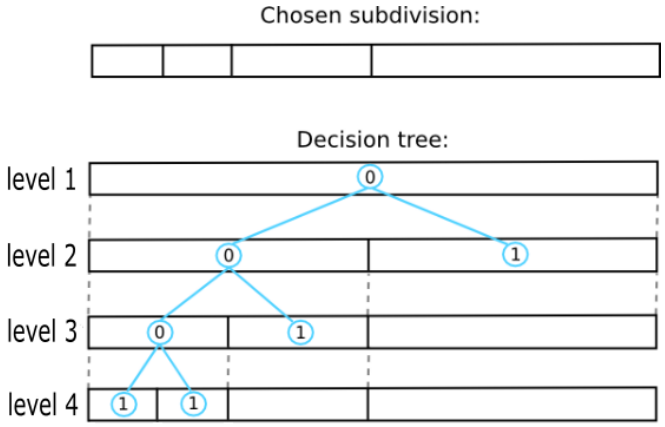


Fig. 7.8 Signaling of the subbands structure: from level 1 downwards, we transmit the labels of the nodes in the binary decision tree.

7.2.3 Image codec for SDCT-AM and SDCT-BT

When using SDCT-AM and SDCT-BT, we need to encode three different types of information: the transform coefficients, the rotation angles, and the subband subdivision. To code the transform coefficients, we perform an uniform quantization and then we code the quantized coefficients using an adaptive bit plane arithmetic coding.

To code the rotation angles, we fix $q_\theta = 8$ quantization levels for the angles, uniformly set in $[0, \pi]$ for both SDCT-AM and SDCT-BT. Then, we use $\log_2 q_\theta = 3$ bit to transmit each rotation angle. We do not perform any compression on the angles, as their distribution, as observed in our tests, does not exhibit an evident compressibility. In order to improve the compression performance, as future work we may consider a non-uniform angle quantization.

Regarding the subband subdivision, the two proposed algorithms present two different encoding methods, as explained in the previous section. SDCT-AM requires $(\lceil \log_2 p \rceil)s$ bits, where $p = \frac{n(n-1)}{2}$, and s is the number of subbands.

As also done in [21], we take into account 1 more bit for each block to declare whether we are applying the directional method or the classical DCT.

Algorithm 4 SDCT-BT

Initialize: $k = 0$, $\hat{\theta} = (\theta_0, \theta_0, \ldots, \theta_0)$ (i.e. 1 subband), $\widehat{J} = J(\hat{\theta})$
for $k = 1 \ldots, \lfloor \log_2 p \rfloor$ **do**
  for each subband s **do**
    Split s into two sets of equal length
    $\theta = \widehat{\theta}$
    Sequentially, for each set $\mathcal{S}$,

$$\theta_j = \omega \quad \text{for all } j \in \mathcal{S}$$

    where $\omega = \arg\min_{x \in \mathrm{Q}_\theta} J$ (found via exhaustive search)
    if $J(\theta) < \widehat{J}$ **then**
      $\widehat{J} = J(\theta)$
      $\widehat{\theta} = \theta$
      the two sets are accepted as new subbands
    end if
  end for
  if no split is performed at the current level k **then**
    break
  end if
end for

7.2.4 Experimental results

In this section, we evaluate the performance of the proposed SDCT-AM and SDCT-BT methods and compare them to the state-of-the-art directional transforms. We perform an objective comparison computing the PSNR and a subjective comparison evaluating the SSIM index [88]. At the end of the section, we also propose some considerations and experiments about a possible future implementation of the SDCT in the HEVC standard.

We test SDCT-AM (Algorithm 3) and SDCT-BT (Algorithm 4) on some standard grayscale images and on intra-frame prediction errors. For the prediction errors, we use HEVC to generate intra-frame prediction residuals on the first frame of few test video sequences. For both images and residual frames, we use different block sizes $n \times n$ with $n \in \{8, 16, 32\}$. We compare their performance against the classical DCT, the Directional DCT [21] and the SDCT with only one rotation angle per block (SDCT-1), as proposed in

the previous section of this chapter. Then, we also show a brief comparison between wavelets and SDCT.

For the DDCT and the SDCT-1, we code the transform coefficients using the same method used for SDCT-AM and SDCT-BT (see Sec. 7.2.3); in addition to the bitrate of the coefficients, we count 3 bit per block to transmit the chosen angle and one additional bit to signal if we are using the directional method or the classical DCT. Regarding the wavelets, we use CDF 9/7 wavelets and we code the transform coefficients with the same method used for the other transforms.

For all our simulations, we consider $q_\theta = 8$ angles uniformly set in $[0, \pi]$, as explained in Section 7.2.3. We initialize both SDCT-AM and SDCT-BT with one single angle, testing all 8 possible initializations and eventually choosing the best one. For SDCT-BT, the maximum number of iterations is set by $\lfloor \log_2 p \rfloor$, while for SDCT-AM we get a stationary point in very few iterations (less than 10).

For SDCT-AM, we need to select the parameter α defined in (7.2). As we do not know $\mathcal{R}_{\hat{\mathbf{x}}}$ and $\|\hat{\mathbf{x}}\|_0$ in advance, we employ the values of $\mathcal{R}_{\hat{\mathbf{x}}}$ and $\|\hat{\mathbf{x}}\|_0$ estimated by the classical DCT, multiplied by 2 (we observe in fact that slight overestimation is more safe).

Objective comparison

In Tables 7.1 and 7.2, we summarize our performance results in terms of average gain in PSNR compared to DCT, evaluated through the Bjontegaard metric [66].

In Table 7.1, the comparison is performed on eight classical grayscale images (House, Barbara, Boat, Lena, Aerial 5.1.10 [89], Stream and Bridge 5.2.10 [89], Couple 4.1.02 [89], Airplane F16 4.2.05 [89]; color images have been converted to grayscale). The gains obtained by DDCT and SDCT-1 are similar, and decrease as the block size increases. An inverse behavior characterizes SDCT-AM and SDCT-BT, which generally improve using larger blocks. For blocks 8×8, the four methods are quite similar, while for large blocks SDCT-AM and SDCT-BT are definitely preferable than DDCT and SDCT-1. The PSNR gain ranges from 0.3 dB to nearly 1 dB.

Image	block size	DDCT	SDCT-1	SDCT-AM	SDCT-BT
House	8×8	0.325	0.382	0.406	0.432
256×256	16×16	0.274	0.335	0.636	0.563
	32×32	0.312	0.259	0.718	0.603
Barbara	8×8	0.285	0.288	0.328	0.321
512×512	16×16	0.153	0.195	0.507	0.392
	32×32	0.074	0.093	0.567	0.448
Boat	8×8	0.238	0.271	0.330	0.301
512×512	16×16	0.105	0.160	0.499	0.338
	32×32	0.043	0.076	0.565	0.392
Lena	8×8	0.349	0.347	0.375	0.378
512×512	16×16	0.260	0.252	0.578	0.460
	32×32	0.170	0.129	0.624	0.519
Aerial	8×8	0.343	0.490	0.476	0.572
256×256	16×16	0.132	0.297	0.512	0.720
	32×32	0.017	0.143	0.455	0.985
Stream	8×8	0.394	0.417	0.442	0.476
512×512	16×16	0.165	0.256	0.547	0.559
	32×32	0.046	0.119	0.522	0.736
Couple	8×8	0.239	0.294	0.341	0.326
256×256	16×16	0.140	0.223	0.570	0.456
	32×32	0.066	0.114	0.620	0.630
F16	8×8	0.286	0.417	0.404	0.459
512×512	16×16	0.198	0.340	0.620	0.632
	32×32	0.094	0.181	0.631	0.729

Table 7.1 Average gain in PSNR with respect to DCT measured with Bjontegaard metric (tests on images)

Prediction residual	block size	DDCT	SDCT-1	SDCT-AM	SDCT-BT
RaceHorses	8×8	0.401	0.443	0.431	0.477
416×240	16×16	0.249	0.313	0.461	0.625
	32×32	0.119	0.164	0.354	0.827
RaceHorses	8×8	0.407	0.431	0.455	0.459
832×480	16×16	0.228	0.278	0.527	0.549
	32×32	0.125	0.138	0.461	0.776
BasketballPass	8×8	0.322	0.381	0.503	0.415
416×240	16×16	0.200	0.235	0.619	0.502
	32×32	0.120	0.133	0.606	0.652
PartyScene	8×8	0.468	0.368	0.335	0.388
832×480	16×16	0.283	0.235	0.307	0.451
	32×32	0.138	0.122	0.243	0.549
ChinaSpeed	8×8	0.613	0.391	0.431	0.382
1024×768	16×16	0.486	0.312	0.565	0.477
	32×32	0.289	0.150	0.491	0.527
Keiba	8×8	0.207	0.380	0.455	0.435
416×240	16×16	0.117	0.226	0.507	0.546
	32×32	0.078	0.098	0.470	0.770
Keiba	8×8	0.267	0.331	0.471	0.367
832×480	16 ×16	0.157	0.205	0.580	0.419
	32×32	0.068	0.086	0.543	0.510
Kristen&Sara	8×8	0.265	0.264	0.417	0.273
1280×720	16×16	0.217	0.220	0.607	0.396
	32×32	0.124	0.129	0.644	0.529

Table 7.2 Average gain in PSNR with respect to DCT measured with Bjontegaard metric (tests on intra-prediction errors)

In Table 7.2, prediction errors are considered on eight different videos. The behavior is similar to that appreciated for images in Table 7.1: the gain obtained by SDCT-AM and SDCT-BT with respect to DDCT and SDCT-1 is more consistent as the block size increases. In this case, the PSNR gain ranges from 0.3 dB to 0.8 dB.

From the results we can see that the performance of SDCT-AM and SDCT-BT are similar. In certain cases (such as Boat or Barbara), SDCT-AM outperforms SDCT-BT. Instead, in other cases the performance of SDCT-AM slightly decreases using larger block sizes, while that of SDCT-BT always increases. This happens mostly with prediction errors and with textured images (such as Aerial), for which the non-regularity may require a higher number of subbands. In such frameworks, SDCT-AM is penalized as it uses a larger number of bits to signal the subbands structure if compared to SDCT-BT.

In Fig. 7.9, we depict the RD curves concerning the image Airplane F16, for $n = 8, 16, 32$. For $n = 16, 32$, SDCT-AM and SDCT-BT turn out to be better than the state-of-the-art methods.

Comparison with wavelets

For still image compression, coding schemes based on wavelets have achieved significantly better performance compared to DCT-based compression methods [90]. Instead in video coding, wavelet-based compression methods have not shown significant performance gains versus DCT-based methods [91]. In our work we consider both images and videos, but our focus is mainly on video compression and a possible future implementation of the SDCT in a video compression standard. For this reason, we use as main benchmark the DCT, that is the core transform of most video coding standards. However, we also present a comparison between wavelets and SDCT on a few sample images.

In Table 7.3 we show a comparison between SDCT and wavelets for $n = 16, 32, 64$. We evaluate the performance of SDCT-AM and SDCT-BT in terms of average gain in PSNR compared to wavelets. As we can see from the results, when the dimension of the block is small, SDCT-AM and SDCT-BT show a significant quality gain. Instead, at larger block size the wavelets usually outperform both SDCT-AM and SDCT-BT. It is interesting to point out that

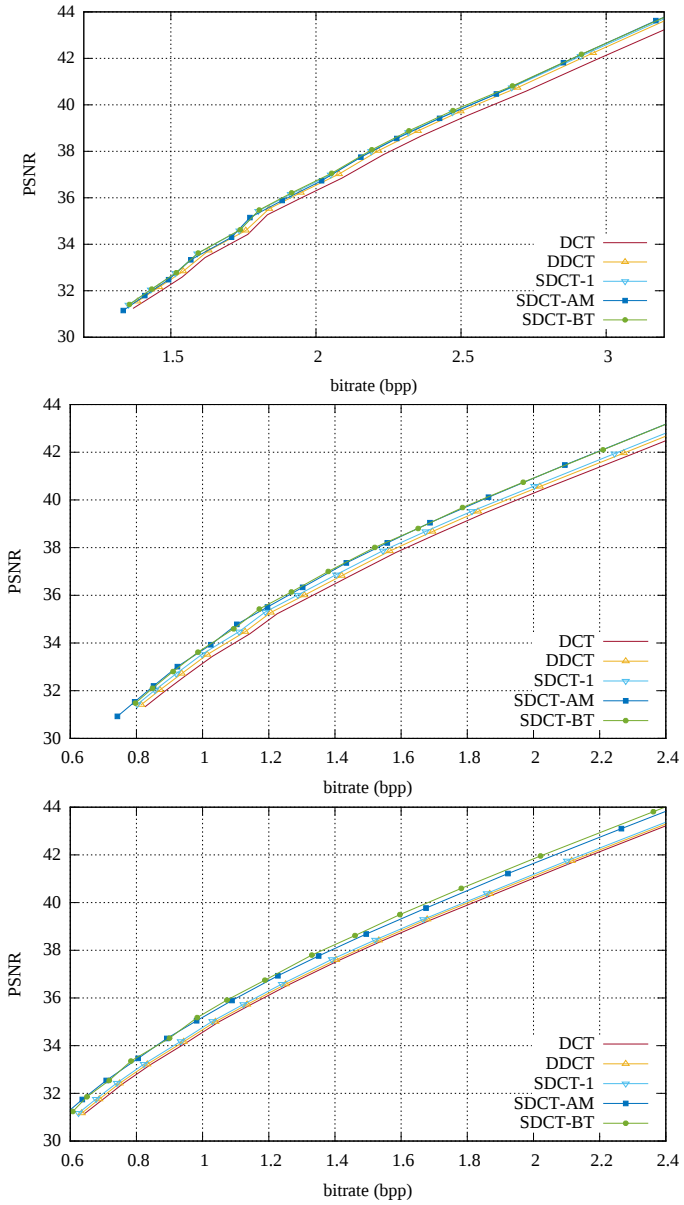


Fig. 7.9 RD performance comparison for the image Airplane F16 using different block sizes: from top to bottom, $n = 8, 16, 32$

in the test Aerial with $n = 64$ SDCT-BT outperforms the wavelets, which in turn outperform the classical DCT. This demonstrates that sometimes the improvement obtained by SDCT is significant to make the DCT approach more efficient than other approaches.

Prediction residual	block size	SDCT-AM	SDCT-BT	DCT
Boat	16×16	1.858	1.702	1.359
512×512	32×32	1.589	1.384	0.985
	64×64	-2.183	-1.862	-3.199
Aerial	16×16	1.724	1.519	1.012
256×256	32×32	1.161	1.690	0.714
	64×64	-0.432	0.294	-0.886
Stream	16×16	1.265	1.272	0.718
512×512	32×32	1.043	1.234	0.495
	64×64	-0.802	-0.232	-1.364

Table 7.3 Average gain in PSNR with respect to wavelets measured with Bjontegaard metric

Subjective comparison

Since the PSNR is not always a good representation of the visual quality, we also compute the SSIM index in order to evaluate the perceived quality. The results for the image Barbara are shown in Fig. 7.10. Also in this case, we can see that when we use smaller blocks the performance of the three directional methods are very similar, instead when the block size increases the SDCT clearly outperforms the other methods.

In Fig. 7.11, we show a detail of F16 (block size 64×64, 0.8 bpp) in which a visual improvement can be observed in SDCT-AM and SDCT-BT with respect to DCT.

Future applications

To conclude the experimental section, we propose some observations and tests regarding possible future applications of SDCT. In particular, we investigate the possibility to implement efficiently the proposed SDCT in the HEVC standard.

In HEVC, the core transform is DCT [92] [93]. Replacing it with SDCT is then expected to produce a performance improvement. A test implementation of SDCT within HEVC is beyond the purpose of this paper and is left for future work. However, it is worth mentioning that HEVC uses an integer

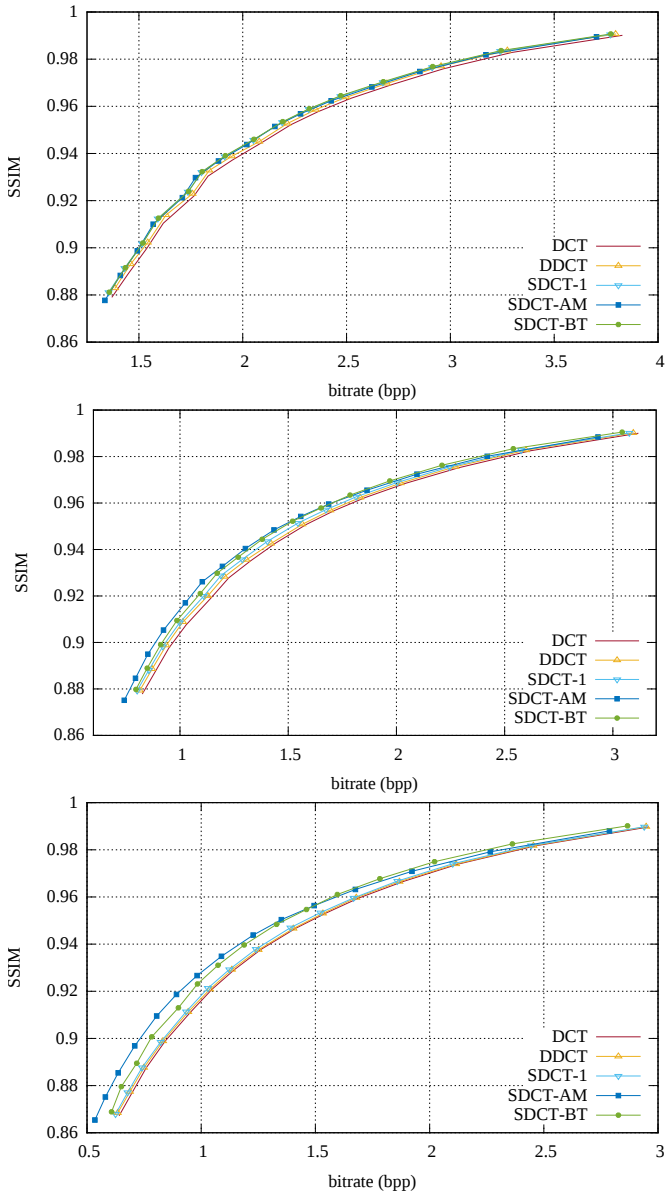
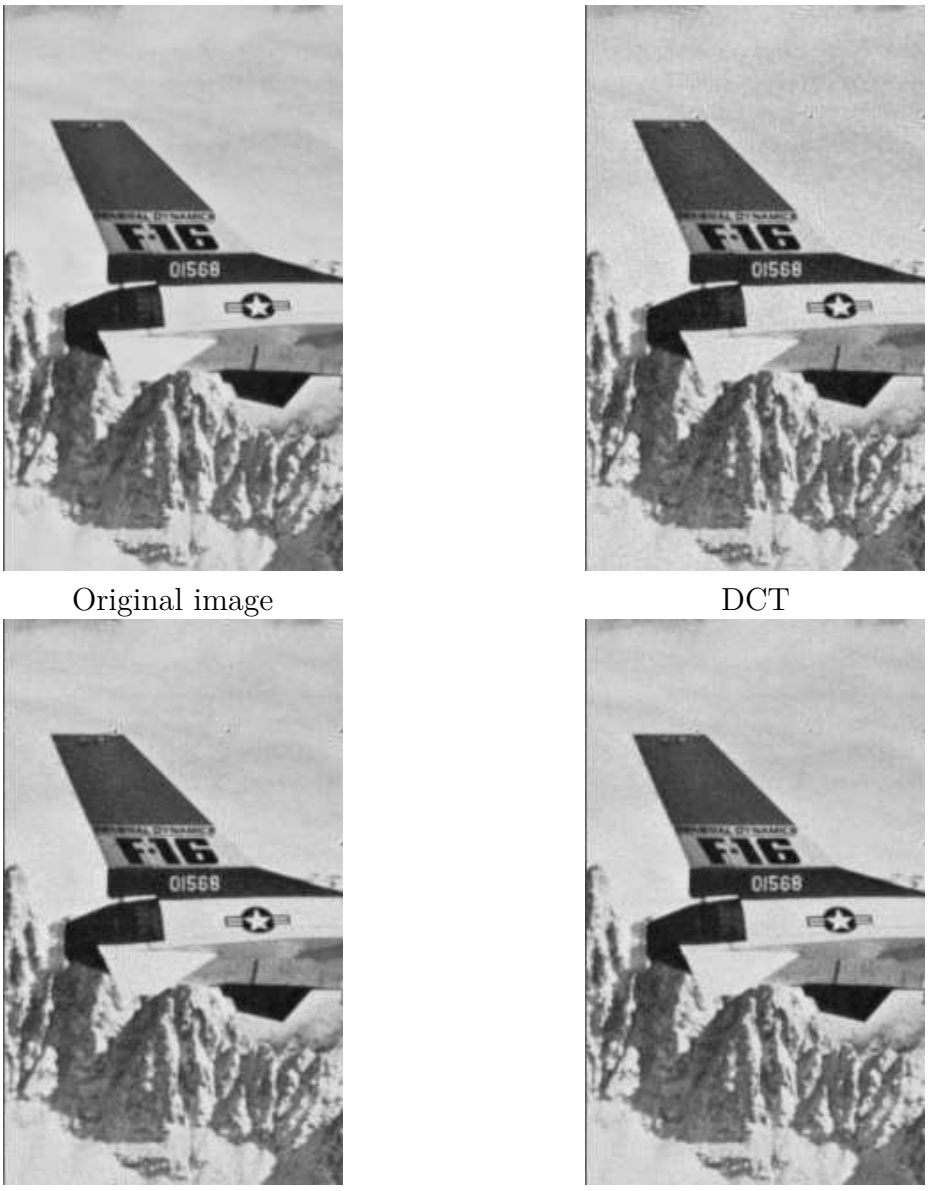


Fig. 7.10 SSIM performance comparison for the image Airplane F16 using different block sizes: from top to bottom, $n = 8, 16, 32$

version of DCT, i.e. an approximate DCT that can be stored using only integer values [93]. This clearly has memory advantages, but involves a not exactly orthogonal transform. For this reason, we have tested SDCT using the same integer approximation in order to evaluate the possible drawbacks. To compute the integer approximation of the proposed SDCT we have used equation (6.6), where the DCT coefficients are computed using the integer DCT defined in



Original image DCT

SDCT-AM SDCT-BT

Fig. 7.11 Visual comparison on Airplane F16 image (block size 64×64, 2 bpp): at the same bpp, DCT is more spotted than SDCT-AM and SDCT-BT.

HEVC. The obtained results (of which we show just some samples in Table 7.4) are in line with the previous non-integer approach. Moreover, Table 7.5 shows that in mostly all cases the proposed SDCT is chosen in a significant number of blocks.

Prediction residual	block size	integer SDCT-AM	integer SDCT-BT
RaceHorses	8×8	0.429	0.476
416×240	16×16	0.458	0.613
	32×32	0.353	0.830
BasketballPass	8×8	0.498	0.413
416×240	16×16	0.615	0.494
	32×32	0.609	0.642
Keiba	8×8	0.452	0.431
416×240	16×16	0.505	0.538
	32×32	0.472	0.754

Table 7.4 Integer SDCT for HEVC: average gain in PSNR with respect to integer DCT measured with Bjontegaard metric

This is a first step that suggests the possibility to implement efficiently an integer SDCT in the HEVC standard.

7.3 Subspace-Sparsifying Steerable DCT

In this section, we propose a new directional DCT based on the SDCT, called Subspace-Sparsifying SDCT (S^3DCT), that utilizes a high number of rotation angles, in order to better compact the energy of the signal. By properly matching the angles to the imge block content, the proposed S^3DCT yields a matrix of transform coefficients that is triangular. In this way, the number of the coefficients to be transmitted to the decoder is nearly halved, resulting in a significant coding gain. We have integrated the S^3DCT into a fully fledged image codec, showing that the new transform significantly outperforms existing conventional and directional DCTs.

The main idea of the S^3DCT is to choose among all the possible rotations that can be performed the one that provides the sparsest coefficients representation.

For each eigenspace with multiplicity 2, we can find in an analytical way the rotation angle that compacts all the energy in one coefficient, zeroing out the other one. Let $\mathbf{x} \in \mathbb{R}^{n^2}$ be the original (vectorized) image block, given

Prediction residual	block size	integer SDCT-AM	integer SDCT-BT
RaceHorses	8×8	50%	47%
416×240	16×16	56%	71%
	32×32	38%	84%
BasketballPass	8×8	64%	46%
416×240	16×16	80%	65%
	32×32	77%	78%
Keiba	8×8	57%	44%
416×240	16×16	79%	60%
	32×32	89%	80%

Table 7.5 Percentage of blocks where the SDCT is chosen over the DCT at 40 dB

an eigenvalue $\lambda_{k,l}$ of $L(\mathcal{P}_n \times \mathcal{P}_n)$ with multiplicity 2 and its corresponding eigenvectors $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$, for the given block, the corresponding DCT coefficients are:

$$\hat{x}_{DCT_{k,l}} = {\mathbf{v}^{(k,l)}}^T \mathbf{x},$$

$$\hat{x}_{DCT_{l,k}} = {\mathbf{v}^{(l,k)}}^T \mathbf{x}$$

Then, the angle that provides the sparsest representation is

$$\theta_{k,l} = \arctan \frac{\hat{x}_{DCT_{k,l}}}{\hat{x}_{DCT_{l,k}}}. \tag{7.5}$$

In fact, given ${\mathbf{v}^{(k,l)}}'$ and ${\mathbf{v}^{(l,k)}}'$, which are obtained rotating $\mathbf{v}^{(k,l)}$ and $\mathbf{v}^{(l,k)}$ by $\theta_{k,l}$ as described in (8.4), the new transform coefficients are

$$\hat{x}_{k,l} = {\mathbf{v}^{(k,l)}}'^T \mathbf{x},$$

$$\hat{x}_{l,k} = {\mathbf{v}^{(l,k)}}'^T \mathbf{x}.$$

Then, from (7.5), we conclude that $\hat{x}_{k,l} = 0$ and all the energy is conveyed to $\hat{x}_{l,k}$, as shown in Fig. 7.12.

In this way, we can define the optimal rotation angle as θ^*, where each component $\theta^*_{k,l}$ is defined as in (7.5). We obtain, then, a transform matrix $V(\theta^*)$ that has the remarkable property that the coefficients matrix obtained is triangular. We call this special case of SDCT as Subspace-Sparsifying SDCT

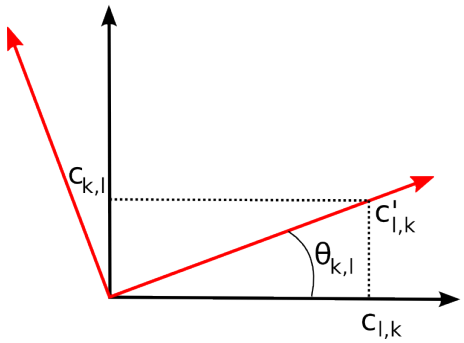


Fig. 7.12 Sparsifying rotation: using the angle defined in (7.3) p transform coefficients are exactly null.

(S^{3}DCT). The main advantage of the S^{3}DCT is that we know a priori that we do not need to transmit p coefficients, since they are forced to be null. In this way, we nearly halve the number of coefficients that have to be sent to the decoder, while, on the other hand, we have to transmit also the rotation angles used. However, the number of rotation angles to transmit can be reduced, since many DCT coefficients (mostly in the higher frequencies) are null. In this case, we can avoid to rotate the corresponding eigenspace and, therefore, to transmit the rotation angles. In this way, we transmit only the rotation angles of the coefficients that are not quantized to zero, reducing significantly the number of angles to transmit. Another advantage of the S^{3}DCT with respect of the DCT lies in the fact that the angle for each subspace can be computed analytically using (7.5); therefore, one does not need to perform an exhaustive search of the optimal angle, leading to significantly reduced complexity.

7.3.1 Rate-Distortion Optimization

As pointed out in the previous section, when we use the S^{3}DCT we have also to take into account the cost of transmitting the rotation angles corresponding to the coefficients that are not null. Therefore, the total rate should take into account both the rate due to the coefficient transmission $\mathcal{R}_{\hat{\mathbf{x}}}$ and the rate due to the angles transmission $\mathcal{R}_\theta$. For this reason, the S^{3}DCT is not necessarily optimal in rate-distortion terms, i.e., if $\mathcal{R}_\theta$ is too high, the RD cost of the S^{3}DCT may be higher than the RD cost of the DCT. In order to address this, we introduce a rate-distortion optimization problem aiming to minimize the

Algorithm 5 Subspace-Sparsifying SDCT (S^3DCT)

Input: $\mathbf{x} \in \mathbb{R}^{n^2}$: (vectorized) image block;
Perform the 2D DCT: $\hat{\mathbf{x}}_{DCT} = V\mathbf{x}$;
Compute the optimal rotation θ^* using (7.5);
$V(\theta*) = R(\theta*)V$;
Perform the S^3DCT: $\hat{\mathbf{x}} = V(\theta^*)\mathbf{x}$
if $J_{\text{DCT}} < J_{\text{S}^3\text{DCT}}$ **then**
    Choose the 2D DCT;
else
    Choose the S^3DCT;
end if

RD cost J

$$\min J, \qquad \text{where } J = \mathcal{D} + \gamma\mathcal{R},$$

where $\mathcal{D}$ and $\mathcal{R}$ respectively are the distortion and rate terms, and the Lagrangian RD cost J is minimized for a particular value of the Lagrangian multiplier γ. For each block, we evaluate the RD cost of the DCT and the one of the S^3DCT and we perform the S^3DCT only when its RD cost is lower than that of the DCT, i.e. $\mathcal{D}_{\text{DCT}} + \lambda\mathcal{R}_{\text{DCT}} > \mathcal{D}_{\text{S}^3\text{DCT}} + \lambda(\mathcal{R}_{\hat{\mathbf{x}}} + \mathcal{R}_\theta)$. Algorithm 5 describes the entire procedure used for S^3DCT.

7.3.2 Encoder

When we use the S^3DCT, we need to transmit to the decoder both the transform coefficients and the rotation angles. To code the coefficients, we first quantize the transform coefficients using the JPEG quantization table. Then, we perform an adaptive bit plane arithmetic coding of the quantized coefficients.

In order to transmit the rotation angles, we first perform an uniform quantization of the angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Therefore, after having computed the optimal rotation θ^* using (7.5), we perform a quantization of the angles obtaining θ^*_q. Then, we use as transform matrix $V(\theta^*_q)$. Since the angles are quantized, the transform may not perfectly sparsify the subspaces, in this case we force to zero the coefficients in the triangular part of the matrix that is not transmitted. From the experiments, we have seen that we obtain good performance even if we use a coarse angle quantization, as shown in the first

	Average gain in PSNR (dB)			Percentage of bitrate saving		
Image	S^3DCT	SDCT-1	DDCT	S^3DCT	SDCT-1	DDCT
Barbara 4×4	0.805	0.174	0.806	6.252	1.35	5.889
Barbara 8×8	0.626	0.276	0.450	5.974	2.636	4.260
Barbara 16×16	0.373	0.198	0.163	4.202	1.811	2.237
Boat 4×4	0.559	0.106	0.413	5.660	1.016	3.952
Boat 8×8	0.466	0.225	0.233	5.779	2.769	2.782
Boat 16×16	0.338	0.133	0.093	4.599	1.815	1.222
Lena 4×4	0.758	0.167	0.474	7.632	1.831	4.784
Lena 8×8	0.585	0.318	0.344	7.929	4.384	4.741
Lena 16×16	0.382	0.208	0.224	6.064	3.336	3.597
House 4×4	0.577	0.128	0.325	6.033	1.190	2.840
House 8×8	0.510	0.280	0.336	6.406	3.172	3.922
House 16×16	0.339	0.174	0.265	4.892	2.303	3.546

Table 7.6 Average gain in PSNR and percentage of bitrate saving measured with the Bjontegaard metric.

section of this chapter. For our experiments, we use a quantization on 8 angles. Therefore, we take into account in the bitrate 3 bits for each rotation angle used and also 1 bit per block to indicate if we use the S^3DCT or the DCT. We leave as future work an analysis of possible efficient compression methods for the angles.

Only the angles corresponding to nonzero coefficients in the upper-triangular part of the S^3DCT coefficients matrix are actually transmitted. It is important to underline that the decoder automatically identifies which subspaces correspond to the received rotation angles, since angles are meaningful only for the coefficients that are not quantized to zero.

7.3.3 Experimental results

To evaluate the proposed method, we use several standard grayscale images. We test our method using different block sizes: 4×4, 8×8 and 16×16. We compare our proposed S^3DCT with the RD-optimized transform mode selection against the classical 2D DCT, the Directional DCT (DDCT) proposed in [21] and the Steerable DCT with one angle (SDCT-1) described in the first section

Bitrate	4×4	8×8	16×16
1 bpp	92%	75%	57%
1.5 bpp	83%	67%	48%
2 bpp	77%	66%	49%

Table 7.7 Percentage of blocks where the S^3DCT is chosen over the DCT for the image Lena.

of this chapter. To obtain a fair comparison, we use the same quantization table and coding technique for all these methods.

Fig. 7.13 shows the rate-distortion curves obtained using the image Lena. More results are shown in Table 7.6 where we use the Bjontegaard metric [66] to compute the average gain in PSNR and the percentage of bitrate saving of the S^3DCT, the DDCT and the SDCT-1 compared to the classical DCT. Instead, in Table 7.7 we show at different bitrate the percentage of blocks where the S^3DCT is chosen over the DCT.

From these results, we can see that the proposed method outperforms the DCT and the other two directional transforms and the average gain over the DCT increases as the block size decreases reaching an average quality gain of approximately 0.65 dB with 4×4 blocks. This is due to the fact that with larger blocks the angles quantization has more influence, reducing the sparsifying capability of the transform. For this reason, we note that the performance of the S^3DCT can be improved even more, especially on large blocks, by introducing an efficient compression methods for the angles. It is also important to underline that the gain of the S^3DCT increases at high bitrate.

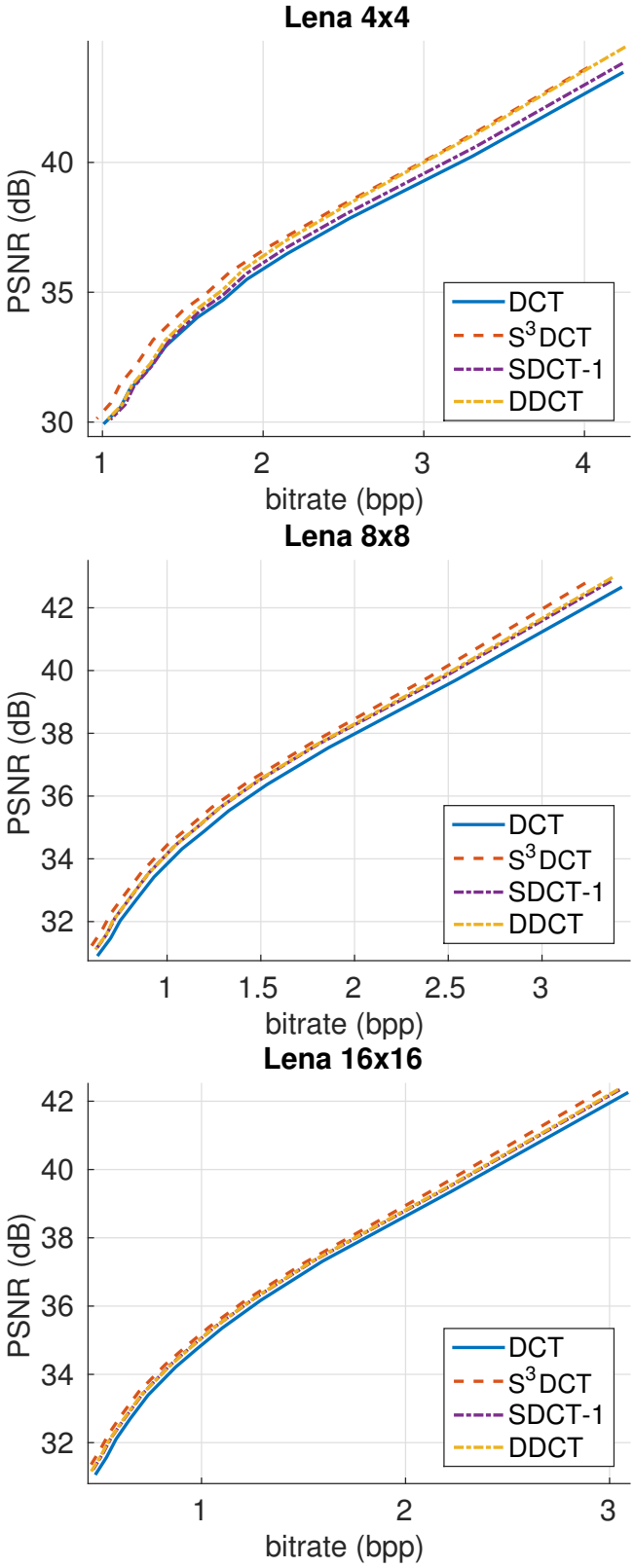


Fig. 7.13 RD comparison between the three directional methods and the DCT.

Chapter 8

Steerable Discrete Fourier Transform

The discrete Fourier transform (DFT) is one of the most important tools in digital signal processing. It enables us to analyze, manipulate, and synthesize signals and it is now used in almost every field of engineering [94]. In the past, some generalizations of the Fourier transform have been presented, such as the short time Fourier transform [95] and the fractional Fourier transform (also called angular Fourier transform) [96] [97]. The short time Fourier transform subdivides the signal into narrow time intervals in order to obtain simultaneous information on time and frequency. Instead, the fractional Fourier transform, and its discrete version called discrete rotational Fourier transform [98], can be interpreted as a rotation on the time-frequency plane. As already discussed in the previous chapters, recently the concept of a graph Fourier transform (GFT) has been introduced in [99]; this new transform generalizes the traditional Fourier analysis to the graph domain.

In Chapter 6, the relationship between the graph Fourier transform and grid graphs has been exploited to define a new directional 2D-DCT [38] that can be steered in a chosen direction. In this chapter, we extend this concept and present a new generalization of the DFT, called steerable discrete Fourier transform (SDFT). The proposed SDFT can be defined in one or two dimensions (unlike the steerable DCT which can be defined only in the 2D case). In 1D,

Part of the work described in this chapter has been previously published in G. Fracastoro, E. Magli, Steerable Discrete Fourier Transform, *IEEE Signal Processing Letters*, 2017.

we start from the definition of the GFT of a cycle graph and we obtain a new transform, the 1D-SDFT, by rotating the 1D-DFT basis. The 1D-SDFT can be interpreted as a rotation of the basis vectors on the complex plane. Instead, in the 2D case we use the GFT of a toroidal grid graph to introduce the new 2D-SDFT, which can be obtained by rotating the 2D-DFT basis. The 2D-SDFT may represent a rotation on the two-dimensional Euclidean space. Since the DFT is used in a wide range of applications, the SDFT represents an interesting generalization that could be applied in various fields, including e.g. filtering, signal analysis, or even multimedia encryption where parametrized versions of common transforms have been used for security purposes [100, 101]. We also show that the SDFT is related to other well-known transforms, such as the Fourier sine and cosine transforms and the Hilbert transform.

8.1 SDFT - 1D case

The forward one dimensional discrete Fourier transform (1D-DFT) of the signal $\mathbf{x} \in \mathbb{R}^N$ can be computed in the following way

$$\hat{\mathbf{x}}_k = \sum_{n=0}^{N-1} \mathbf{x}_n \mathrm{e}^{-i\frac{2\pi kn}{N}}.$$

We can write it in matrix form

$$\hat{\mathbf{x}} = V\mathbf{x},$$

where $V_{kn} = \mathrm{e}^{-i\frac{2\pi kn}{N}} = \rho_k^n$. $V \in \mathbb{C}^{N\times N}$ is the 1D-DFT matrix and it has the following property.

Theorem 8.1 (Theorem 5.1 [39])**.** *The rows of the DFT matrix are eigenvectors of any circulant matrix.*

Let us now consider an undirected cycle graph $\mathcal{C}_N$ with N vertices. As we have said in Chapter 2, the Laplacian matrix of a cycle graph is circulant and it is well known that a valid set of eigenvectors for any circulant matrix is the set of DFT matrix rows, then the 1D-DFT is a valid GFT for $\mathcal{C}_N$ (i.e. $\Psi^H = V$,

where H denotes the Hermiatian transpose[1]). However, repeated eigenvalues are present in the spectrum of $L(\mathcal{C}_N)$, because the following property holds

$$\lambda_k = \lambda_{N-k} \tag{8.1}$$

where λ_k is the k-th eigenvalue of $L(\mathcal{C}_N)$ with $k = 1, 2, ..., \frac{N}{2} - 1$ [102]. The eigenvalues λ_k can be computed in the following way [102]:

$$\lambda_k = l_0 + l_1\rho_k + l_2\rho_k^2 + \cdots + l_{N-1}\rho_k^{N-1} = 2 - 2\cos\frac{2\pi k}{N}, \tag{8.2}$$

for $k = 0, 1, 2, ..., N-1$. In addition, $L(\mathcal{C}_N)$ has N orthogonal eigenvectors $\{\mathbf{v}^{(k)}\}$ [102]

$$\mathbf{v}^{(k)} = \begin{bmatrix} 1 \\ \rho_k \\ \rho_k^2 \\ \vdots \\ \rho_k^{n-1} \end{bmatrix}, \tag{8.3}$$

for $k = 0, 1, 2, ..., N-1$.

From (8.1) and (8.2), we can state that, if N is even, λ_0 and $\lambda_{\frac{N}{2}}$ have algebraic multiplicity 1, instead all the other eigenvalues have algebraic multiplicity 2 with $\lambda_k = \lambda_{N-k}$, where $1 \leq k \leq \frac{N}{2} - 1$. Since the eigenvectors are orthogonal, the geometric multiplicity is equal to the algebraic multiplicity. This means that the dimension of the eigenspaces corresponding to λ_k where $1 \leq k \leq \frac{N}{2} - 1$ is 2, then the vector basis of the 1D-DFT is not the only possible eigenbasis of $L(\mathcal{C}_N)$.

We can then introduce the following corollary, whose proof follows from the discussion above and is omitted for brevity.

Corollary 8.1. *The graph Fourier transform of a cycle graph $\mathcal{C}_N$ may be equal to the 1D-DFT, but it is not the only possible graph Fourier transform of a cycle graph.*

We now proceed to define the 1D-SDFT. Given an eigenvalue λ_k of $L(\mathcal{C}_N)$ with multiplicity 2 and the two corresponding 1D-DFT vectors $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(N-k)}$,

[1]Since in this chapter we use complex numbers, the GFT matrix is defined using the Hermitian transpose instead of the transpose, which was used in all the other chapters.

we can define any other possible basis of the eigenspace coerresponding to λ_k as the result of a rotation of $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(N-k)}$

$$\begin{bmatrix} \mathbf{v}^{(k)'} \\ \mathbf{v}^{(N-k)'} \end{bmatrix} = \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(k)} \\ \mathbf{v}^{(N-k)} \end{bmatrix}, \tag{8.4}$$

where θ_k is an angle in $[0, 2\pi]$.

For every λ_k where $1 \leq k \leq \frac{N}{2} - 1$, we can rotate the corresponding eigenvectors as shown in (8.4). In the 1D-DFT matrix, the pairs $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(N-k)}$ are replaced with the rotated ones $\mathbf{v}'^{(k)}$ and $\mathbf{v}'^{(N-k)}$ obtaining a new transform matrix $V(\theta) \in \mathbb{C}^{N \times N}$ called 1D-SDFT. The vector $\theta \in \mathbb{R}^p$ contains all the rotation angles used and its length is $p = \frac{N}{2} - 1$. The new transform matrix $V(\theta)$ can be written as

$$V(\theta) = VR(\theta), \tag{8.5}$$

where $V = V(0) \in \mathbb{C}^{N \times N}$ is the 1D-DFT matrix and $R(\theta) \in \mathbb{R}^{N \times N}$ is the rotation matrix, whose structure is defined so that, for each pair of eigenvectors, it performs the rotation as defined in (8.4). It is important to underline that the choice of the eigenvector pairs is given by the analysis of the eigenvalue multiplicity. In this way, the transform defined in (8.5) is still the graph transform of a cycle graph.

Equation (8.5) shows that the SDFT can be obtained by applying the rotation described by $R(\theta)$ to the output of the standard DFT, that can be easily computed using the FFT.

From a geometrical point of view, (8.4) represents a rotation in the complex plane. Given a real-valued signal $\mathbf{x} \in \mathbb{R}^N$, its DFT coefficients $\hat{\mathbf{x}}$ have the following symmetry property [94]

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^*_{N-k},$$

where $1 \leq k \leq \frac{N}{2} - 1$ and the "$*$" symbol denotes conjugation. Then, using the rotation in (8.4) we can break this symmetry. For example, if we perform a rotation by $\frac{\pi}{4}$, we can completely separate the real part and the imaginary part. In fact, given $\mathbf{v}^{(k)'}$ and $\mathbf{v}^{(N-k)'}$, which are obtained rotating $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(N-k)}$

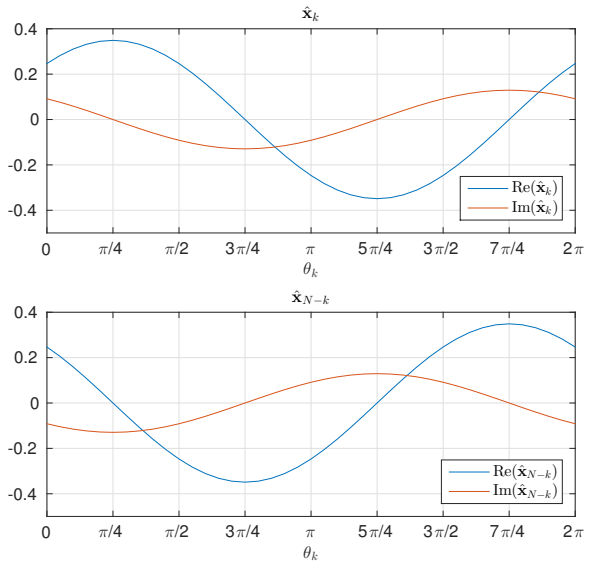


Fig. 8.1 An example of a pair of coefficients $\hat{\mathbf{x}}'_k$ and $\hat{\mathbf{x}}'_{N-k}$ as a function of the rotation angle $\theta \in [0, 2\pi]$.

by $\frac{\pi}{4}$ as in (8.4), the new transform coefficients are

$$\begin{aligned}
\begin{bmatrix} \hat{\mathbf{x}}'_k \\ \hat{\mathbf{x}}'_{N-k} \end{bmatrix} &= \begin{bmatrix} \mathbf{v}^{(k)\prime} \\ \mathbf{v}^{(N-k)\prime} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(k)} \\ \mathbf{v}^{(N-k)} \end{bmatrix} \mathbf{x} \\
&= \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{N-k} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}^*_k \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{2}\ \mathrm{Re}(\hat{\mathbf{x}}_k) \\ -i\sqrt{2}\ \mathrm{Im}(\hat{\mathbf{x}}_k) \end{bmatrix},
\end{aligned}$$

where $i = \sqrt{-1}$. As an example, Fig. 8.1 shows a plot of a pair of coefficients $\hat{\mathbf{x}}'_k$ and $\hat{\mathbf{x}}'_{N-k}$ as a function of the rotation angle $\theta_k \in [0, 2\pi]$. For both coefficients, we can clearly see that the absolute values of the real and imaginary part are inversely proportional. Moreover, when $\theta_k = (2t+1)\frac{\pi}{4}$ with $t = 0, 1, 2, 3$ one coefficient is a real value and the other one is a pure imaginary value. Finally, it is important to underline that the rotation described in (8.4) preserves the total energy of the coefficients in the eigenspace

$$|\hat{\mathbf{x}}_k|^2 + |\hat{\mathbf{x}}_{N-k}|^2 = |\hat{\mathbf{x}}'_k|^2 + |\hat{\mathbf{x}}'_{N-k}|^2.$$

8.1.1 Relationships of the 1D-SDFT to other transforms

We have already shown that if $\theta = 0$ the 1D-SDFT is equal to the 1D-DFT. Instead if $\theta = \frac{\pi}{4}$, it is interesting to point out that for $1 \leq k \leq \frac{N}{2} - 1$ we have that $\hat{\mathbf{x}}_k = \sqrt{2}\mathbf{x}_k^{\cos}$, where $\mathbf{x}_k^{\cos}$ is the k-th coefficient of the Fourier cosine transform [103]. Analogously, for $\frac{N}{2} + 1 \leq k \leq N - 1$ we have that $\hat{\mathbf{x}}_k = -i\sqrt{2}\mathbf{x}_k^{\sin}$, where $\mathbf{x}_k^{\sin}$ is the k-th coefficient of the Fourier sine transform [103]. In turn, the Fourier cosine and sine transform are highly related respectively to the DCT and DST [103].

Moreover, we can also relate the 1D-SDFT with the Hilbert transform [104]. In fact, given a signal $\mathbf{x}$ it can be proved that

$$\mathcal{H}(\mathbf{x}) = \operatorname{Im}\left(\tilde{V}\left(\frac{\pi}{4}\right)^H V\left(-\frac{\pi}{4}\right)\mathbf{x}\right), \tag{8.6}$$

where $\mathcal{H}(\mathbf{x})$ is the Hilbert transform of $\mathbf{x}$ and $\tilde{V}\left(\frac{\pi}{4}\right)$ is the 1D-SDFT corresponding to the improper rotation [105]

$$\begin{bmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} \end{bmatrix},$$

and $\operatorname{Re}\left(\tilde{V}\left(\frac{\pi}{4}\right)^H V\left(-\frac{\pi}{4}\right)\mathbf{x}\right) = (\mathbf{v}^{(0)^T}\mathbf{x})\mathbf{v}^{(0)} + (\mathbf{v}^{\left(\frac{N}{2}\right)^T}\mathbf{x})\mathbf{v}^{(\frac{N}{2})}$.

These relationships are interesting because they may open the way to new generalizations of these transforms.

8.2 SDFT - 2D case

In the two dimensional case, the 2D-DFT of a signal $X \in \mathbb{R}^{N_1 \times N_2}$ can be computed as follows

$$\hat{X}_{kl} = \sum_{n=0}^{N_1-1}\sum_{m=0}^{N_2-1} X_{mn} \mathrm{e}^{-i2\pi\left(\frac{l}{N_1}n + \frac{k}{N_2}m\right)},$$

in matrix form we can write it as

$$\hat{\mathbf{x}} = W\mathbf{x},$$

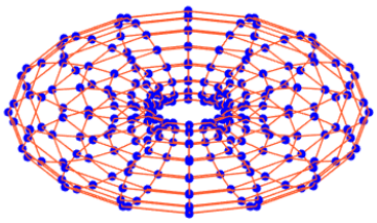


Fig. 8.2 A toroidal grid graph $\mathcal{T}_{16,16}$

where $\mathbf{x} \in \mathbb{R}^{N_1 N_2}$ is the vectorized signal X and $W \in \mathbb{C}^{N_1 N_2 \times N_1 N_2}$ is the 2D-DFT matrix, which is defined in the following way

$$W_{ts} = \mathrm{e}^{-i2\pi\left(\frac{l}{N_1}n + \frac{k}{N_2}m\right)} = \rho_l^n \rho_k^m,$$

where $s = mN_1 + n$, $t = kN_1 + l$, $0 \leq l, n \leq N_1 - 1$ and $0 \leq m, k \leq N_2 - 1$.

We now consider a grid graph with periodic boundary conditions that is called toroidal grid graph $\mathcal{T}_{N_1 N_2}$ [106], where $|\mathcal{V}| = N_1 N_2$. An example of a toroidal grid graph is shown in Fig. 8.2. It is known that the toroidal grid graph $\mathcal{T}_{N_1 N_2}$ corresponds to the product graph $\mathcal{C}_{N_1} \times \mathcal{C}_{N_2}$, where $\mathcal{C}_{N_i}$ is a cycle of N_i vertices [107].

We now show that there is a strong connection between 2D-DFT and toroidal grid graph.

Theorem 8.2. *Let $\mathcal{T}_{NN}$ be a toroidal graph, then the 2D-DFT basis is an eigenbasis of $L(\mathcal{T}_{NN})$.*

Proof. Let $\mathbf{v}^{(p)}$ and $\mathbf{v}^{(q)}$, where $0 \leq p, q \leq N - 1$, be the eigenvectors of $\mathcal{C}_N$ corrisponding respectively to the eigenvalues λ_p and λ_q, as defined in (8.3). Then, using Theorem 6.1 we can compute the eigenvector $\mathbf{u}^{(p,q)}$ of $\mathcal{T}_{NN}$ corresponding to the eigenvalue $\mu_{p,q} = \lambda_p + \lambda_q$

$$\mathbf{u}^{(p,q)} = \mathbf{v}^{(p)} \otimes \mathbf{v}^{(q)} = \begin{bmatrix} \mathbf{v}_1^{(p)} \mathbf{v}^{(q)} \\ \mathbf{v}_2^{(p)} \mathbf{v}^{(q)} \\ \vdots \\ \mathbf{v}_{n-1}^{(p)} \mathbf{v}^{(q)} \end{bmatrix} = \begin{bmatrix} \mathbf{v}^{(q)} \\ \rho_p \mathbf{v}^{(q)} \\ \vdots \\ \rho_p^{n-1} \mathbf{v}^{(q)} \end{bmatrix} = \mathbf{w}^{(k)},$$

where $k = pN + q$ and $\mathbf{w}^{(k)}$ is the k-th column vector of the 2D-DFT matrix W. Therefore, the 2D-DFT is an eigenbasis of the Laplacian of $\mathcal{T}_{NN}$ (i.e. $\Psi^H = W$).

□

Since $\mu_{p,q} = \lambda_p + \lambda_q = \mu_{q,p}$ and recalling property (8.1) for the eigenvalues λ_k of $\mathcal{C}_N$, in the spectrum of $L(\mathcal{T}_{NN})$ several repeated eigenvalues are presents:

- The eigenvalues $\mu_{p,q}$ where $1 \leq p, q \leq \frac{N}{2} - 1$ and $p \neq q$ have algebraic multiplicity 8 since $\mu_{p,q} = \mu_{q,p} = \mu_{p,N-q} = \mu_{N-q,p} = \mu_{N-p,q} = \mu_{q,N-p} = \mu_{N-p,N-q} = \mu_{N-q,N-p}$.
- The eigenvalues $\mu_{p,p}$ where $1 \leq p \leq \frac{N}{2} - 1$ have algebraic multiplicity 4 since $\mu_{p,p} = \mu_{p,N-p} = \mu_{N-p,p} = \mu_{N-p,N-p}$.
- The eigenvalues $\mu_{p,q}$ where $p = 0, \frac{N}{2}$ and $1 \leq q \leq \frac{N}{2} - 1$ (or $1 \leq p \leq \frac{N}{2} - 1$ and $q = 0, \frac{N}{2}$) have algebraic multiplicity 4 because $\mu_{p,q} = \mu_{q,p} = \mu_{p,N-q} = \mu_{N-q,p}$ $(\mu_{p,q} = \mu_{q,p} = \mu_{N-p,q} = \mu_{q,N-p})$.
- The eigenvalue $\mu_{0,\frac{N}{2}} = \mu_{\frac{N}{2},0}$ has multiplicity 2.
- The eigenvalues $\mu_{0,0}$ and $\mu_{\frac{N}{2},\frac{N}{2}}$ are the only ones with algebraic multiplicity 1.

Since the Kronecker product is not commutative, the eigenvectors $\mathbf{u}^{(p,q)}$ of $\mathcal{T}_{NN}$ are orthogonal. Then, the geometric multiplicity is equal to the algebraic multiplicity. Therefore, the dimension of the eigenspaces corresponding to the repeated eigenvalues is bigger than one. This proves that the 2D-DFT is not the unique eigenbasis for $L(\mathcal{T}_{NN})$ and, thus, the 2D-DFT is not the unique GFT for $\mathcal{T}_{NN}$. Instead, the rotations that exploit the property $\mu_{p,q} = \mu_{N-q,N-p}$ and $\mu_{N-p,q} = \mu_{p,N-q}$ are analog to the ones shown in the 1D case.

As shown above, in the spectrum of $L(\mathcal{T}_{NN})$ many eigenvalues with multiplicity greater than 2 are present. Therefore, it may be possible to define rotations in more than two dimensions. However, these rotations may not have a clear geometrical meaning. For this reason, in the following of this section we restrict our study to rotations in two dimensions that exploit the symmetric property $\mu_{p,q} = \mu_{q,p}$.

Given any vector pair of the 2D-DFT, $\mathbf{u}^{(p,q)}$ and $\mathbf{u}^{(q,p)}$ where $p \neq q$, we can obtain a new pair of eigenvectors of $L(\mathcal{T}_{NN})$ by performing the following rotation

$$\begin{bmatrix} \mathbf{u}^{(p,q)'} \\ \mathbf{u}^{(q,p)'} \end{bmatrix} = \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(p,q)} \\ \mathbf{u}^{(q,p)} \end{bmatrix}, \tag{8.7}$$

where $\theta_{p,q}$ is an angle in $[0, 2\pi]$. Then, analogously to the 1D case, we can define a new transform matrix $V(\theta) \in \mathbb{R}^{N^2 \times N^2}$, called 2D-SDFT, that is obtained by replacing in the 2D-DFT matrix the pairs $\mathbf{u}^{(p,q)}$ and $\mathbf{u}^{(q,p)}$ with the rotated ones $\mathbf{u}^{(p,q)'}$ and $\mathbf{u}^{(q,p)'}$. The vector $\theta \in \mathbb{R}^p$ contains all the angles used and its length is equal to the number of vector pairs, that is $p = \frac{N(N-1)}{2}$. Similarly to the 1D case, also the 2D-SDFT matrix $V(\theta)$ can be computed as in (8.5), where, in this case, $R(\theta) \in \mathbb{R}^{N^2 \times N^2}$ is the rotation matrix whose structure is defined so that, for each pair of vectors, it performs the rotation as defined in (8.7).

Given a signal $\mathbf{x} \in \mathbb{R}^{N \times N}$, we can compute the SDFT coefficients of $\mathbf{x}$ corresponding to the eigenvectors $\mathbf{u}^{(p,q)'}$ and $\mathbf{u}^{(q,p)'}$ in the following way

$$\begin{aligned}
\begin{bmatrix} \hat{\mathbf{x}}'_{p,q} \\ \hat{\mathbf{x}}'_{p,q} \end{bmatrix} &= \begin{bmatrix} \mathbf{u}^{(p,q)'} \\ \mathbf{u}^{(q,p)'} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(p,q)} \\ \mathbf{u}^{(q,p)} \end{bmatrix} \mathbf{x} \\
&= \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{p,q} \\ \hat{\mathbf{x}}_{q,p} \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \mathrm{Re}(\hat{\mathbf{x}}_{p,q}) + \mathrm{Im}(\hat{\mathbf{x}}_{p,q}) \\ \mathrm{Re}(\hat{\mathbf{x}}_{p,q}) + \mathrm{Im}(\hat{\mathbf{x}}_{p,q}) \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \mathrm{Re}(\hat{\mathbf{x}}_{p,q}) \\ \mathrm{Re}(\hat{\mathbf{x}}_{p,q}) \end{bmatrix} + i \begin{bmatrix} \cos\theta_{p,q} & \sin\theta_{p,q} \\ -\sin\theta_{p,q} & \cos\theta_{p,q} \end{bmatrix} \begin{bmatrix} \mathrm{Im}(\hat{\mathbf{x}}_{p,q}) \\ \mathrm{Im}(\hat{\mathbf{x}}_{p,q}) \end{bmatrix}.
\end{aligned}$$

Therefore, we can state that, from a geometrical point of view, (8.7) performs separately a rotation of the real and imaginary part in the 2D Euclidean space. Then, by applying (8.7) the total energy of the real and imaginary part of the coefficient pair remains unchanged, that is

$$\mathrm{Re}(\hat{\mathbf{x}}_{p,q})^2 + \mathrm{Re}(\hat{\mathbf{x}}_{q,p})^2 = \mathrm{Re}(\hat{\mathbf{x}}'_{p,q})^2 + \mathrm{Re}(\hat{\mathbf{x}}'_{q,p})^2,$$

$$\mathrm{Im}(\hat{\mathbf{x}}_{p,q})^2 + \mathrm{Im}(\hat{\mathbf{x}}_{q,p})^2 = \mathrm{Im}(\hat{\mathbf{x}}'_{p,q})^2 + \mathrm{Im}(\hat{\mathbf{x}}'_{q,p})^2,$$

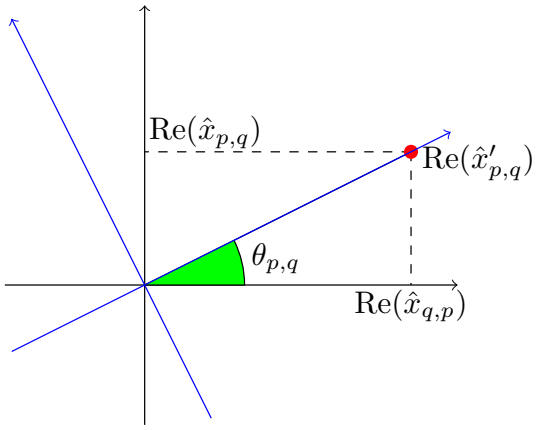


Fig. 8.3 Rotation of the 2D-DFT vector pair $\mathbf{u}^{(p,q)}$ and $\mathbf{u}^{(q,p)}$.

but it is possible to unbalance the energy of the real and imaginary part of each coefficient. For example, we can compact all the energy of the real part in one coefficient, zeroing out the other one. In fact, given the pair of DFT coefficients $\hat{\mathbf{x}}_{p,q}$ and $\hat{\mathbf{x}}_{q,p}$ we rotate the pair of corresponding DFT vectors $\mathbf{u}^{(p,q)}$ and $\mathbf{u}^{(q,p)}$ as in (8.7) by an angle defined as follows

$$\theta_{p,q} = \arctan \frac{\mathrm{Re}(\hat{\mathbf{x}}_{q,p})}{\mathrm{Re}(\hat{\mathbf{x}}_{p,q})}.$$

Then, we obtain that $\mathrm{Re}(\hat{\mathbf{x}}'_{q,p}) = 0$ and all the energy of the real part of this coefficient pair is conveyed to $\hat{\mathbf{x}}'_{p,q}$, as shown in Fig. 8.3.

8.3 Applications of the SDFT

In this section we discuss possible applications of the SDFT.

The 1D-SDFT can be useful for signal analysis and processing. For example, it can be used for easily filtering the even/odd component of a signal. In fact, if we rotate the pairs of vectors $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(N-k)}$ by $\frac{\pi}{4}$, we can design a filter that, convolved with the input signal, retains only the first (last) $\frac{N}{2}$ coefficients and outputs the even (odd) signal component, as shown in Fig. 8.4 where we obtain as output of the filter the even component of the input signal. We can also easily filter the even or odd component of specific frequencies. Analogously in 2D, we can perform the same filtering operation by rotating by $\frac{\pi}{4}$ the pairs

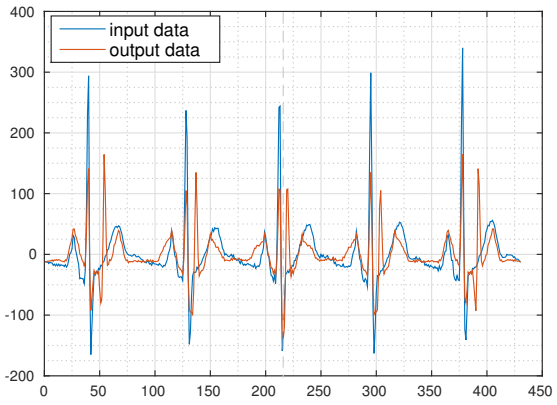


Fig. 8.4 Example of filtering the even component of a real signal.

of vectors $\mathbf{u}^{(p,q)}$ and $\mathbf{u}^{(N-p,N-q)}$ and the pairs $\mathbf{u}^{(p,N-q)}$ and $\mathbf{u}^{(N-p,q)}$. This filtering operation could be useful for signal representation, as in [108].

Moreover, in (8.6) we have already shown that the 1D-SDFT may be used to perform the Hilbert transform, this could be useful for computing the local phase and amplitude, that is used in many applications, such as edge detection [109] and image feature extraction [110].

The 1D-SDFT can be applied also in multimedia encryption problems. In this field, several works use parametrized versions of common transforms for security purposes [100, 101]. Since the SDFT is a parametrized version of the DFT, one can use the parameter $\theta \in \mathbb{R}^p$ as a secret key. More specifically, given a signal $\mathbf{x} \in \mathbb{R}^p$ we can obtain $\hat{\mathbf{x}} = V(\theta)\mathbf{x}$ and then consider the first $\frac{N}{2}$ components of $\hat{\mathbf{x}}$ as the encrypted signal. Given θ, it is possible to reconstruct $\hat{\mathbf{x}}$ and then we can obtain the original signal $\mathbf{x}$ by applying the inverse SDFT. We can also consider the SDFT as a keyed transform basis that can be used for compressed sensing-based cryptography [111, 112].

The applications presented in this section are just a few examples of possible applications of the SDFT, but the SDFT could be of interest for a wide range of fields, such as array signal processing, phase retrieval and magnetic resonance imaging.

Chapter 9

Conclusions

In this thesis we explored the use of adaptive transforms in image and video compression. More specifically, we investigated two different methods for building an adaptive transform. Firstly, we exploited the graph representation of an image in order to propose new compression methods. Since graph-based compression methods have the disadvantage that the graph has to be transmitted to decoder, we focused on the problem of developing new graph construction methods that provide a graph that is, at the same time, an efficient representation of the image and easy to transmit to the decoder. To address this problem, we used different approaches. First, we developed an innovative method of graph construction that employs edge quantization and a new edge prediction technique. Then, in order to find the graph that provides the best tradeoff between its transmission cost and the quality of the corresponding transform, we applied a graph learning approach defining a new graph learning problem targeted for image compression. Moreover, we also explored the use of graph transform in conjunction with computer vision tools, such as superpixels. Exploiting these techniques, we obtained new graph-based image compression methods that provide a significant quality gain over the classical DCT.

In the second part of this thesis, we exploited the graph Fourier transform in order to introduce a novel directional transform, called Steerable Discrete Cosine Transform (SDCT). The proposed SDCT can be steered in any chosen direction and we can use more complex steering patterns than a single pure rotation. We investigated the applications of the SDCT in image and video compression

developing a few algorithms based on this new directional transform. Since the SDCT admits more than one rotation angle per block, these algorithms explore different methods for the choice of the rotation angles. The obtained results show that the SDCT can outperform the classical DCT both for image and video compression.

Along the same lines, we also presented a new generalization of the DFT, called Steerable Discrete Fourier Transform (SDFT), that can be defined in 1D and 2D. Since the DFT is used in a wide range of applications, the SDFT could be of interest in many fields, including e.g. filtering, signal analysis or even multimedia encryption.

9.1 Future work

In this thesis, we have described new methods for designing adaptive transforms. However, there are still some paths which can be followed and which may lead to interesting research problems. In the follow we describe some future work which may be interesting to further investigate.

- *Implementation of the SDCT in the HEVC video coding standard:* In Chapter 7, we have already tested the SDCT with the integer approximation of the DCT that is used in HEVC. The obtained results show that the integer SDCT provides an improvement in performance with respect to the integer DCT. For this reason, it would be interesting to implement the proposed SDCT in HEVC in order to see if the SDCT can provide an improvement in the standard performance.
- *Application of the SDCT in image interpolation:* In this thesis we introduced a new directional transform and we showed its application in image and video compression application. It would be also interesting to investigate other possible applications of the SDCT, such as image interpolation. In this case we could exploit the transform directionality in order to obtain an interpolation that better matches the directional characteristics of the image.
- *Extension of the SDCT to 3D:* In this work we have defined the SDCT in 2D. However, it is possible to define a steerable DCT also in 3D.

In this case, the 3D-SDCT could have interesting applications in video compression. Differently from the 2D-SDCT, with the 3D-SDCT we may define two different steering directions: one in the spatial dimension and one in the temporal dimension.

- *Dictionary learning for graph representation:* In Chapter 4, we show that the graph of an image can be treated as a signal that lies on the unweighted dual graph. Whilst this representation turned out to be an effective way to represent the graph structure, it could be useful to investigate other possible representation for the graph, like dictionary-based techniques. In this case, we may avoid to impose the strong assumption that consecutive edges have similar weights.

- *Graph-based transform for intra-prediction residual coding:* In Chapter 3, 4 and 5 we have presented three different methods for image compression. However, in these chapters we did not consider the use of intra-prediction. In future, it would be interesting to investigate the possible application of these techniques to intra-prediction residuals.

- *Applications of the SDFT:* In Chapter 8, we have introduced a new generalization of the DFT. In future, it could be interesting to investigate possible applications of the SDFT. As we previously discussed, the SDFT may be of interest in a wide range of fields, such as signal analysis and representation, array signal processing, phase retrieval and magnetic resonance imaging. Moreover, we have shown that the SDFT could be applied also in multimedia encryption problems.

References

[1] K. Sayood, *Introduction to data compression.* Newnes, 2012.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] A. Fuldseth, G. Bjontegaard, S. Midtskogen, T. Davies, and M. Zanaty, "Thor video codec," 2016, https://tools.ietf.org/html/draft-fuldseth-netvc-thor-02.

[4] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec vp9-an overview and preliminary results," in *Proc. Picture Coding Symposium (PCS)*, 2013, pp. 390–393.

[5] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[6] G. Shen, W. S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *Picture Coding Symposium (PCS)*, 2010, pp. 2808–2811.

[7] W. S. Kim, S. K. Narang, and A. Ortega, "Graph based transforms for depth video coding," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012, pp. 813–816.

[8] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 419–433, 2015.

[9] W. Hu, G. Cheung, and A. Ortega, "Intra-prediction and generalized graph fourier transform for image coding," *IEEE Signal Process. Lett.*, vol. 22, no. 11, 2015.

[10] Y. H. Chao, A. Ortega, and S. Yea, "Graph-based lifting transform for intra-predicted video coding," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 1140–1144.

[11] H. E. Egilmez, A. Said, Y.-H. Chao, and A. Ortega, "Graph-based transforms for inter predicted video coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 3992–3996.

[12] K.-S. Lu and A. Ortega, "Symmetric line graph transforms for inter predictive video coding," in *Proc. Picture Coding Symposium (PCS)*, 2016.

[13] S. K. Narang, Y. H. Chao, and A. Ortega, "Graph-wavelet filterbanks for edge-aware image processing," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, 2012, pp. 141–144.

[14] S. K. Narang, Y.-H. Chao, and A. Ortega, "Critically sampled graph-based wavelet transforms for image coding," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2013, pp. 1–4.

[15] Y.-H. Chao, A. Ortega, W. Hu, and G. Cheung, "Edge-adaptive depth map coding with lifting transform on graphs," in *Proc. Picture Coding Symposium (PCS)*, 2015, pp. 60–64.

[16] E. Martínez-Enríquez and A. Ortega, "Lifting transforms on graphs for video coding," in *Proc. Data Compression Conference (DCC)*, 2011, pp. 73–82.

[17] E. Martínez-Enríquez, F. Díaz-de María, and A. Ortega, "Video encoder based on lifting transforms on graphs," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 3509–3512.

[18] E. Martinez-Enriquez, J. Cid-Sueiro, F. Diaz-De-Maria, and A. Ortega, "Directional transforms for video coding based on lifting on graphs," *IEEE Trans. Circuits Syst. Video Technol.*, 2016.

[19] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. IEEE International Conference on Computer Vision (CVPR)*, 2003, pp. 10–17.

[20] J. Xu, B. Zeng, and F. Wu, "An overview of directional transforms in image coding." in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 3036–3039.

[21] B. Zeng and J. Fu, "Directional discrete cosine transforms - a new framework for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 305–313, 2008.

[22] C. L. Chang and B. Girod, "Direction-adaptive partitioned block transform for image coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 145–148.

[23] F. Kamisli and J. S. Lim, "Transforms for the motion compensation residual," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009, pp. 789–792.

[24] R. A. Cohen, S. Klomp, A. Vetro, and H. Sun, "Direction-adaptive transforms for coding prediction residuals," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 185–188.

[25] A. Drémeau, C. Herzet, C. Guillemot, and J. Fuchs, "Sparse optimization with directional dct bases for image compression," in *Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2010, pp. 1290–1293.

[26] C. Yeo, Y. H. Tan, Z. Li, and S. Rahardja, "Mode-dependent transforms for coding directional intra prediction residuals," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 545–554, 2012.

[27] H. Xu, J. Xu, and F. Wu, "Lifting-based directional DCT-like transform for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1325–1335, 2007.

[28] Y. Ye and M. Karczewicz, "Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 2116–2119.

[29] M. Budagavi and M. Zhou, "Orthogonal MDDT and mode dependent DCT," *ITU-T Q*, vol. 6, 2010.

[30] H. Yang, J. Zhou, and H. Yu, "Simplified MDDT (SMDDT) for intra prediction residual," *Doc. JCTVC-B039, MPEG-H/JCT-VC*, 2010.

[31] A. Tanizawa, J. Yamaguchi, T. Shiodera, T. Chujoh, and T. Yamakage, "Improvement of intra coding by bidirectional intra prediction and 1 dimensional directional unified transform," *Doc. JCTVC-B042, MPEG-H/JCT-VC*, 2010.

[32] O. G. Sezer, R. Cohen, and A. Vetro, "Robust learning of 2-d separable transforms for next-generation video coding," in *Proc. IEEE Data Compression Conference (DCC)*, 2011, pp. 63–72.

[33] X. Zhao, L. Zhang, S. Ma, and W. Gao, "Video coding with rate-distortion optimized transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 1, pp. 138–151, 2012.

[34] O. G. Sezer, O. Harmanci, and O. G. Guleryuz, "Sparse orthonormal transforms for image compression," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 149–152.

[35] O. G. Sezer, O. G. Guleryuz, and Y. Altunbasak, "Approximation and compression with sparse orthonormal transforms," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2328–2343, 2015.

[36] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *Proc. ICML workshop on statistical relational learning and its connections to other fields*, vol. 15, 2004, pp. 67–68.

[37] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[38] G. Strang, "The discrete cosine transform," *SIAM review*, vol. 41, no. 1, pp. 135–147, 1999.

[39] L. Grady and J. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*. Springer, 2010.

[40] W. Hu, X. Li, G. Cheung, and O. Au, "Depth map denoising using graph-based transform and group sparsity," in *Proc. IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013, pp. 1–6.

[41] J. Pang, G. Cheung, A. Ortega, and O. C. Au, "Optimal graph laplacian regularization for natural image denoising," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 2294–2298.

[42] A. Kheradmand and P. Milanfar, "A general framework for kernel similarity-based image denoising," in *Proc. IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013, pp. 415–418.

[43] J. Pang and G. Cheung, "Graph laplacian regularization for inverse imaging: Analysis in the continuous domain," *arXiv preprint arXiv:1604.07948*, 2016.

[44] X. Liu, D. Zhai, D. Zhao, G. Zhai, and W. Gao, "Progressive image denoising through hybrid graph laplacian regularization: a unified framework," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1491–1503, 2014.

[45] Y. Wang, A. Ortega, D. Tian, and A. Vetro, "A graph-based joint bilateral approach for depth enhancement," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 885–889.

[46] W. Hu, G. Cheung, X. Li, and O. C. Au, "Graph-based joint denoising and super-resolution of generalized piecewise smooth images," in *Proc IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2056–2060.

[47] A. Kheradmand and P. Milanfar, "A general framework for regularized, similarity-based image restoration," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5136–5151, 2014.

[48] X. Liu, G. Cheung, X. Wu, and D. Zhao, "Inter-block consistent soft decoding of jpeg images with sparsity and graph-signal smoothness priors," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 1628–1632.

[49] W. Hu, G. Cheung, and M. Kazui, "Graph-based dequantization of block-compressed piecewise smooth images," *IEEE Signal Process. Lett.*, vol. 23, no. 2, pp. 242–246, 2016.

[50] C. Zhang and D. Florêncio, "Analyzing the optimality of predictive transform coding using graph-based models," *IEEE Signal Process. Lett.*, vol. 20, no. 1, pp. 106–109, 2013.

[51] M. Black, G. Sapiro, D. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 21–432, 1998.

[52] M. Hidane, O. Lezoray, and A. Elmoataz, "Lifting scheme on graphs with application to image representation," in *Proc. IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013, pp. 431–434.

[53] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990.

[54] G. Sullivan, "Decoder inference of optimal reconstruction values for DZ+UTQ quantization of laplacian source random variables," in *Proc. 16th JVT meeting, JVT-P111*, 2005.

[55] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped motion prediction for depth video coding using arithmetic edge coding," *IEEE Trans. Image Process.*, vol. 23, no. 11, pp. 4696–4708, 2014.

[56] E. Pavez and A. Ortega, "Generalized laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2016, pp. 6350–6354.

[57] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.

[58] E. Pavez, H. E. Egilmez, Y. Wang, and A. Ortega, "GTT: Graph Template Transforms with applications to image coding," in *Proc. Picture Coding Symposium (PCS)*, 2015, pp. 199–203.

[59] J. Gallier, "Elementary spectral graph theory applications to graph clustering using normalized cuts: a survey," *arXiv preprint arXiv:1311.2492*, 2013.

[60] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[61] X. Liu, G. Cheung, and X. Wu, "Joint denoising and contrast enhancement of images using graph laplacian operator," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 2274–2278.

[62] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 1027–1042, 1998.

[63] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 13, 2016.

[64] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[65] I. Rotondo, G. Cheung, A. Ortega, and H. E. Egilmez, "Designing sparse graphs via structure tensor for block transform coding of images," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2015, pp. 571–574.

[66] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.

[67] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, 2009.

[68] J. Wang and X. Wang, "VCells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1241–1247, 2012.

[69] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, 2012.

[70] P. Krajcevski and D. Manocha, "SegTC: Fast texture compression using image segmentation," *Proc. High Performance Graphics*, pp. 71–77, 2014.

[71] N. Brewer, L. Wang, N. Liu, and L. Cheng, "User-driven lossy compression for images and video," in *Proc. International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2009, pp. 346–351.

[72] G. Sharma, W. Wu, and E. Dalal, "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Res. Appl.*, vol. 30, no. 1, pp. 21–30, 2005.

[73] F. Verdoja and M. Grangetto, "Fast superpixel-based hierarchical approach to image segmentation," in *Proc. International Conference on Image Analysis and Processing (ICIAP)*, 2015, pp. 364–374.

[74] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electron. Lett.*, vol. 44, no. 13, pp. 800–801, 2008.

[75] *Rec. T. 82 & ISO/IEC 11544: 1993*, 1st ed., 1993.

[76] Y. K. Liu and B. Žalik, "An efficient chain code with huffman coding," *Pattern Recognition*, vol. 38, no. 4, pp. 553–557, 2005.

[77] H. Sanchez-Cruz and R. M. Rodriguez-Dagnino, "Compressing bilevel images by means of a three-bit chain code," *Optical engineering*, vol. 44, no. 9, 2005.

[78] F. Verdoja and M. Grangetto, "Efficient representation of segmentation contours using chain codes," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[79] R. Franzen, "Kodak lossless true color image suite," 2013, http://r0k.us/graphics/kodak/index.html.

[80] M. Blaser, C. Heithausen, and M. Wien, "Segmentation-based partitioning for motion compensated prediction in video coding," in *Proc. Picture Coding Symposium (PCS)*, 2016.

[81] S. Milani and G. Calvagno, "Segmentation-based motion compensation for enhanced video coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 1649–1652.

[82] R. Merris, "Laplacian matrices of graphs: a survey," *Linear algebra and its applications*, vol. 197, pp. 143–176, 1994.

[83] ——, "Laplacian graph eigenvectors," *Linear algebra and its applications*, vol. 278, no. 1, pp. 221–236, 1998.

[84] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 1996.

[85] Y. K. Kim, Z. He, and S. K. Mitra, "A novel linear source model and a unified rate control algorithm for H. 263/MPEG-2/MPEG-4," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001, pp. 1777–1780.

[86] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, 1998.

[87] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, no. 5-6, pp. 629 – 654, 2008.

[88] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[89] USC-SIPI, "Image database, volume 3: Miscellaneous," http://sipi.usc.edu/database/database.php?volume=misc.

[90] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice*. Springer Science & Business Media, 2012, vol. 642.

[91] T. Sikora, "Trends and perspectives in image and video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 6–17, 2005.

[92] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (HEVC) standard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1029–1041, 2013.

[93] M. Wien, *High Efficiency Video Coding*. Springer, 2015.

[94] R. G. Lyons, *Understanding digital signal processing*. Pearson Education, 2004.

[95] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[96] L. B. Almeida, "The fractional fourier transform and time-frequency representations," *IEEE Trans. Signal Process.*, vol. 42, no. 11, pp. 3084–3091, 1994.

[97] ——, "An introduction to the angular Fourier transform," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, 1993, pp. 257–260.

[98] B. Santhanam and J. H. McClellan, "The discrete rotational Fourier transform," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 994–998, 1996.

[99] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.

[100] A. Pande and J. Zambreno, "The secure wavelet transform," in *Embedded Multimedia Security Systems*. Springer, 2013, pp. 67–89.

[101] G. Unnikrishnan, J. Joseph, and K. Singh, "Optical encryption by double-random phase encoding in the fractional fourier domain," *Opt. Lett.*, vol. 25, no. 12, pp. 887–889, 2000.

[102] G. J. Tee, "Eigenvectors of block circulant and alternating circulant matrices," *N. Z. J. Math.*, vol. 36, pp. 195–211, 2007.

[103] A. D. Poularikas, *Transforms and applications handbook.* CRC press, 2010.

[104] F. R. Kschischang, "The hilbert transform," *University of Toronto*, 2006.

[105] D. Salomon, *Computer graphics and geometric modeling.* Springer Science & Business Media, 2012.

[106] J. H. Park and I. Ihm, "Many-to-many two-disjoint path covers in cylindrical and toroidal grids," *Discrete Appl. Math.*, vol. 185, pp. 168–191, 2015.

[107] F. Ruskey and J. Sawada, "Bent hamilton cycles in d-dimensional grid graphs," *Electron. J. of Combin.*, vol. 10, 2003.

[108] A. Gnutti, F. Guerrini, and R. Leonardi, "Representation of signals by local symmetry decomposition," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2015, pp. 983–987.

[109] P. Kovesi, "Image features from phase congruency," *Videre: Journal of computer vision research*, vol. 1, no. 3, pp. 1–26, 1999.

[110] G. Carneiro and A. D. Jepson, "Phase-based local features," in *Proc. European Conference on Computer Vision (ECCV)*, 2002, pp. 282–296.

[111] T. Bianchi, V. Bioglio, and E. Magli, "Analysis of one-time random projections for privacy preserving compressed sensing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 313–327, 2016.

[112] L. Y. Zhang, K.-W. Wong, Y. Zhang, and J. Zhou, "Bi-level protected compressive sampling," *arXiv preprint arXiv:1406.1725*, 2014.